Cybersecurity

RESEARCH

Open Access

# EPASAD: ellipsoid decision boundary based Process-Aware Stealthy Attack Detector

Vikas Maurya[1*], Rachit Agarwal[1,2], Saurabh Kumar[1] and Sandeep Shukla[1]

**Abstract**

Due to the importance of Critical Infrastructure (CI) in a nation's economy, they have been lucrative targets for cyber attackers. These critical infrastructures are usually Cyber-Physical Systems such as power grids, water, and sewage treatment facilities, oil and gas pipelines, etc. In recent times, these systems have suffered from cyber attacks numerous times. Researchers have been developing cyber security solutions for CIs to avoid lasting damages. According to standard frameworks, cyber security based on identification, protection, detection, response, and recovery are at the core of these research. Detection of an ongoing attack that escapes standard protection such as firewall, anti-virus, and host/network intrusion detection has gained importance as such attacks eventually affect the physical dynamics of the system. Therefore, anomaly detection in physical dynamics proves an effective means to implement defense-in-depth. PASAD is one example of anomaly detection in the sensor/actuator data, representing such systems' physical dynamics. We present EPASAD, which improves the detection technique used in PASAD to detect these micro-stealthy attacks, as our experiments show that PASAD's spherical boundary-based detection fails to detect. Our method EPASAD overcomes this by using Ellipsoid boundaries, thereby tightening the boundaries in various dimensions, whereas a spherical boundary treats all dimensions equally. We validate EPASAD using the dataset produced by the TE-process simulator and the C-town datasets. The results show that EPASAD improves PASAD's average recall by 5.8% and 9.5% for the two datasets, respectively.

**Keywords** Intrusion detection system, Critical infrastructure security, Industrial control system, Machine learning

## Introduction

Critical infrastructures (CIs) are mostly Cyber-Physical Systems (CPS) with few exceptions (such as telecommunication, financial services, and Agriculture) that facilitate and boost societal and economical operations. Some examples of CIs include infrastructure supporting supply of natural gas, water treatment and supply, electricity generation and renewable energy, food production and distribution, transportation, healthcare, and goods and services. The architecture of a CI is layered- an industrial control system (ICS - also known as cyber-physical systems (CPS)), Supervisory Control and Data Acquisition systems (SCADA), and Process Control Systems (PCS or Distributed Control Systems (DCS)) monitor and control the infrastructure (Cardenas et al. 2011). These high-level designs of supervisory systems are often networked with Programmable Logic Controllers (PLCs). PLCs are industrial computational devices coupled with sensors and actuators to control physical processes by communicating usually with SCADA. The SCADA system is comprised of numerous intrusion detection systems (IDSs) that monitor physical processes or network data generated by sensors and actuators on a regular basis and generate an alarm if the system behaves abnormally.

Minor damage to a CI may lead to catastrophe and significantly impacts public safety, economy, and daily life demands. With the rise of the Internet and connected

*Correspondence:
Vikas Maurya
vikasmr@cse.iitk.ac.in
[1] Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, kanpur, India
[2] Merkle Science, Bangalore, India

things, CIs have become more vulnerable to cyber-attacks. A state's vested interests further escalate this. In the past, there have been numerous cases where cyber-criminals successfully infiltrated CIs. For example, an attack on the Iranian power plant in 2009 was conducted using Stuxnet (Falliere et al. 2010) malware. Other examples of such attacks include (i) attack on a German steel mill in 2014 (Lee et al. 2014) that was conducted using spear-phishing through mails, (ii) attack on the Ukrainian power grid in 2015 which was conducted using spear-phishing via Microsoft doc file affecting ≈225,000 customers (Lee et al. 2016), and (iii) attack on a Saudi petroleum refinery in 2017 using TRITON malware caused the refinery to shut down its operations (Di Pinto et al. 2018). Besides these, there are numerous recent attempts reported by the Center for Strategic and International Studies (CSIS). These include attacks on Indian nuclear plant in 2019, Israel water treatment plant in 2020, and oil and natural gas pipeline companies in USA in 2021 (CSIS 2022).

Thus, the question that motivates us is *how we can secure CIs from such attacks?* Multiple methods are used to answer this and secure CIs. These methods include securing network architecture by adhering to the policies such as network segmentation and segregation, the use of boundary protection devices, and firewall filters between each network segment (Stouffer et al. 2015). However, network security is constantly being breached due to exploitation of vulnerabilities that also include zero-day attacks. Assuming that network security is foolproof and no attacker will break it to cause harm to the ICS is not correct. Only by bypassing the network security attackers do not harm the CI until they perform any malicious activity. When an attacker performs any malicious activity on the CI, it gets reflected in the physical processes (Zheng et al. 2015). The sensors and control behavior associated with attack-targeted devices start to show structural changes in their normal behavior. Usually it happens in direct damage attack (DDA). Such structural changes can be identified to detect an attack. However, an attacker can hide their manipulation within the noise margin. These attacks are known as

stealthy attacks (SA). Such attacks are likely to produce a cascading effect due to the interaction of control loops, eventually causing the control system to fail. Further, an attacker can reduce the impact of SA in such way the sensor produced abnormal structural changes do not deviate much from the normal behavior. We call such attacks as micro-stealthy attacks (MSA) (cf. "Proposed framework: EPASAD" section). These attacks are extremely difficult to detect and evade current state-of-the-art detection techniques. The MSA does not interrupt or fail the control system but slowly degrades the system causing huge losses in terms of money and raw material over an extended period. In this paper, we develop a novel IDS framework whose objective is to detect the most challenging attack category MSA, and quickly detect the SA and DDA to save the CIs from lasting damage.

A process-level intrusion detection system (IDS) continuously monitors the physical process of ICSs. It is deployed over SCADA, whose goal is to detect any abnormal structural changes in the physical process behavior. State-of-the-art approaches categorize process level IDS in two categories: univariate (independent IDS for each sensor) and multivariate (Single IDS for multiple sensor). Among the many popular IDS-based solutions (discussed in "Related works" section ), PASAD (Aoudi et al. 2018) is one of the most promising framework. Yet, PASAD suffers from same drawbacks as mentioned earlier. It fails to detect the micro-stealthy attacks (MSAs), and is delayed in detecting Stealthy Attacks (SAs) and Direct Damage Attacks (DDAs). We consider PASAD as a baseline for the validation of our approach.

An efficient and realistic process level IDS must fulfill following objectives: (i) be capable of detecting an attack before lasting damage, (ii) be secure against evasion attack, (iii) work under noisy environment, (iv) be realistic to build and deploy, (v) have less computational overhead and produce the result for streaming data quickly, and (vi) have lower false alarm rate. Motivated by this, we present an efficient and realistic process level IDS solution called EPASAD (Ellipsoid decision boundary based Process-Aware Stealthy Attack Detector), which addresses above-mentioned objectives. EPASAD is a univariate process-level IDS based on Singular Spectrum Analysis (SSA) (Broomhead and King 1986; Elsner and Tsonis 2013; Golyandina and Korobeynikov 2014; Golyandina et al. 2001; Golyandina and Zhigljavsky 2013; Hassani 2010; Vautard and Ghil 1989), a time series analysis tool. EPASAD is designed to detect any structural changes in a sensor behavior caused due to the presence of an adversary. EPASAD projects raw sensor data into a noise-free lower-dimensional signal subspace to cluster normal data. It uses the distribution of the clustered data to learn an efficient and uniformly tight decision boundary. EPASAD envelops the signal subspace within an ellipsoid decision boundary. After learning the decision boundary, any sensor datum that falls outside is considered abnormal, and an alarm is raised. From the attacker's perspective, in PASAD, it is easier to compromise huge redundant normal space within the decision-boundary and easier to determine the radius using any one projected dimension. However, EPASAD provides a tighter boundary with respective radii in each dimension, necessitating more effort for attackers to determine ellipsoid parameters in each projected dimension to stay

under the radar. This is the novelty that we bring over the existing state-of-the-art approaches.

We consider various attack scenarios present in the two datasets: TE-process and C-town dataset to validate EPASAD. We use TE chemical process simulator to generate two MSAs, three SAs, and two DDA scenarios with the motivation to simulate realistic situations. Our results show that EPASAD successfully detects the MSAs and quickly detects the stealthy and the DDAs when compared to the baseline method PASAD. Further, we consider C-town network datasets to validate our approach on a much larger dataset. Using the dataset, we validate EPASAD using 14 different attacks scenarios that happen over a testing period of 9-months. Each attack present in the C-town dataset is sandwiched between a long-duration normal operation. Testing for such an extended period validates our framework for a realistic scenario. We show that EPASAD is capable of detecting all 14 attacks present in the C-town dataset with a low false alarms rate of 3.7%. Compared to PASAD, EPASAD shows a significant improvement over each attack scenarios. Over PASAD, EPASAD improves the overall recall for all the sensors in the system operating under MSAs, SAs, and DDAs present in the TE-dataset from 7.5% to 17.3%, 50.3% to 54.2%, and 46.2% to 51.0%, respectively. Similarly, in the C-town dataset, EPASAD improves the overall recall from 54.8% to 64.3% for all the 14 attacks present in it. When an attacker attacks a CPS, the behavioral change to anomalous state takes time. But the training data is labeled as "attack" as soon as the attacker engages. This is why the low accuracy appears in both the PASAD and EPASAD. In such low accuracy scenarios, an improvement of even 3.9% (the minimum average gain among all scenarios mentioned above) might appear small from an absolute perspective. From a relative perspective, it is a significant improvement.

In summary, the major contributions of our work are:

- We introduce an attack scenario called Micro-Stealthy Attack (MSA), which although existed but was not studied before and posed detection challenges for current state-of-the-art approaches (cf. "Micro stealthy attack (MSA)" section).
- Our framework called EPASAD provides an efficient and realistic process-level univariate IDS for securing CIs. EPASAD continuously monitors the data stream consisting of sensor measurements for detecting tiny structural changes in the normal behavior hidden within the noise margin.
- We validate EPASAD on MSA and find that EPASAD efficiently detects them. Further, EPASAD significantly improves PASAD without any additional computation overhead. We compare EPASAD with PASAD, using multiple attack scenarios present in the TE-process and C-town dataset.

The rest of the paper is organized as follows: first we discuss the required concepts that form a background knowledge needed to understand EPASAD in "Background" section . Then, in "Attack model" section, we describe the attack model that forms the motivation behind the proposal of the EPASAD framework. In the "Proposed framework: EPASAD" section, we present our proposed framework EPASAD and provide detail of its training, online testing process, and computation cost analysis. In "Validation datasets" section, we describe the generated and the existing datasets used for validation. In "Experiments and results" section , we experimentally validate our method under three subsections and report our results. In "Related works" section , we discuss the related works, mainly highlighting the process-level IDS. Finally, in "Discussion and conclusion" section we conclude our paper along with an in-depth discussion.

## Background

In this section, we discuss the techniques and the concepts that are useful for this work.

### Singular spectral analysis (SSA)

SSA is a non-parametric model-free time series analysis tool with a wide range of applications (Broomhead and King 1986; Elsner and Tsonis 2013; Vautard and Ghil 1989; Golyandina et al. 2001; Mohammad and Nishida 2011), including IDS (Aoudi et al. 2018; Terai et al. 2018; Dong et al. 2017; Moskvina and Zhigljavsky 2003; Golyandina et al. 2001; Mohammad and Nishida 2011). SSA can robustly recover the deterministic pattern of a time series even in the presence of noise. Such aspects of SSA enable us to use it to analyze the noise-free structure of a time series. SSA is also used to identify structural changes in a time series data by learning a projection matrix $\mathbf{P}$ that projects a real-valued noisy subseries into a noise-free signal subspace. However, to do so, only two steps of SSA are sufficient. As our focus is to identify structural changes in normal sensor measurements, here, we only explain these two steps. Note that a summary of the notations/symbols used in this paper are listed in the Table 1. The two steps are:

*Step 1: (Embedding)* This step maps a univariate time series into a trajectory matrix. Let $\mathcal{T} = \{m_1, m_2, \ldots, m_N\}$ be a univariate time series of length $N$ where $m_i \in \mathbb{R}$ is a sensor's measurement collected at the $i$th timestamp. Let $L \in \mathbb{I}$ where $1 < L < N/2$ be called as lag or window-length and $K = N - L + 1$. The SSA arranges the time series $\mathcal{T}$ in the form of a trajectory matrix $\mathbf{M}$ of dimension $L \times K$.

**Table 1** Notations and their description

| Notation | Description |
|---|---|
| $\mathbb{R}$ | Set of Real numbers |
| $\mathbb{I}$ | Set of Integers |
| $m_i$ | $i$th Measurement |
| $\mathbf{M}$ | Trajectory Matrix of size $L \times K$ |
| $m$ | $L$-length lagged vector |
| $M_i$ | A specific lagged vector of length $L$, $i$th column vector of $\mathbf{M}$ or test subsequence for $i$th measurement |
| $c$ | Centroid vector in $\mathbb{R}^L$ |
| $\mathbf{P}$ | Projection matrix |
| $\mathbf{U}$ | Eigen matrix |
| $U_i$ | $i$th Eigenvector |
| $\mathbf{X}$ | A signal subspace matrix of size $R \times K'$ |
| $X_i$ | A specific $R$-length lagged vector in $\mathbb{R}^R$, $i$th column vector of $\mathbf{X}$ or projected test subsequence for $i$th measurement |
| $x$ | A $R$-length lagged vector in signal subspace |
| $w$ | A weight vector in $\mathbb{R}^R$ |
| $\hat{c}$ | Centroid vector in $\mathbb{R}^R$ |
| $\mathcal{D}_t$ | Departure score at timestamp $t$ |
| $\theta_p$ | Threshold of PASAD |
| $\theta_e$ | Threshold of EPASAD |
| $\delta_f(x)$ | Tightness of decision boundary $f(x)$ at a point $x$ |
| $N$ | Length of training subsequence |
| $N'$ | Length of training + validation subsequence |
| $L$ | Lag parameter in $\mathbb{I}$ |
| $R$ | Dimensionality of signal subspace parameter |
| $\epsilon$ | Slack-value parameter |
| $\prod(w)$ | Product of elements of vector $w$ |

$$\mathbf{M} = \begin{bmatrix} m_1 & m_2 & \dots & m_K \\ m_2 & m_3 & \dots & m_{K+1} \\ . & . & & . \\ . & . & & . \\ m_L & m_{L+1} & \dots & m_N \end{bmatrix}$$

A column vector of $\mathbf{M}$ is called the lagged vector where the $i$th $(1 \le i \le K)$ lagged vector $(M_i)$ is defined by $M_i = [m_i, m_{i+1}, \dots, m_{i+L-1}]^T$.

*Step 2: singular value decomposition (SVD)* In this step, SVD of $\mathbf{M}$ is done by using the following four steps: (i) compute a lagged co-variance matrix $\mathbf{M}\mathbf{M}^T$ of dimension $L \times L$, (ii) compute the eigenvalues denoted by $\lambda_1, \lambda_2, \dots, \lambda_L$ and the corresponding eigenvectors denoted by $U_1, U_2, \dots, U_L$, which are arranged according to decreasing magnitude of eigenvalues, (iii) orthonormalize the eigenvectors, and (iv) pick $R$ leading eigenvectors to form eigen matrix $\mathbf{U}$ of dimension $L \times R$, i.e., $\mathbf{U} = [U_1, U_2, \dots, U_R]$. Ignoring the minor and keeping the leading eigenvectors in the eigen matrix $\mathbf{U}$ eliminates the noise and retains the deterministic behavior of a signal subseries. The set of eigenvectors $\{U_1, U_2, \cdots, U_R\}$ are linearly independent, spanning an $R$-dimensional subspace in $\mathbb{R}^L$ (length of vectors in the $R$-dimensional subspace is $L$) called signal subspace. There exists $\mathbf{P} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T = \mathbf{U}\mathbf{U}^T$ (since $\mathbf{U}$ is an orthonormal matrix, then $\mathbf{U}^T\mathbf{U} = \mathbf{I}$) that projects a lagged vector from $L$-dimensional real space to the signal subspace. Let $m \in \mathbb{R}^L$ be a lagged vector, then the projection of $m$, i.e., $\mathbf{P}m \in \mathbb{R}^L$, be a noise-free vector in signal subspace. Note that the notation $m$ is an $L$-length variable lagged vector while $M_i$ is a constant representing $i$th column vector of matrix $\mathbf{M}$.

### PASAD

In Aoudi et al. (2018), the authors describe PASAD, a process-level, univariate, and anomaly-based IDS that monitors ICS process activity in real-time to determine whether the system is operating normally or abnormally. The motivation behind is to detect any aberrant structural change in the physical process to detect stealthy and direct damage attacks.

PASAD leverages from SSA to learn $\mathbf{P} = \mathbf{U}\mathbf{U}^T$. To reduce the computational overhead of PASAD, in Aoudi et al. (2018), the authors proved that an $L-$dimensional lagged vector $m$ projected by $\mathbf{P} = \mathbf{U}^T$ in $\mathbb{R}^R$ preserves the Euclidean distance projected by $\mathbf{P} = \mathbf{U}\mathbf{U}^T$ in $\mathbb{R}^L$, i.e., $||\mathbf{U}\mathbf{U}^T v|| = ||\mathbf{U}^T v||$. The $\mathbf{P} = \mathbf{U}^T$ captures the deterministic behavior of the physical process by projecting an $L$ -dimensional normal subseries onto a lower $R$-dimensional signal subspace. PASAD computes the squared Euclidean from centroid in $R$-dimensional signal space for each streaming test lagged vectors $M_i$ $(i > N)$ called departure score $(\mathcal{D}_i)$ to detect the attack-induced structural changes in the normal behavior. $\mathcal{D}_i$ is defined via Eq. 1, where $\hat{c} = \mathbf{U}^T c$ and $c$ is the mean of column vectors of $\mathbf{X}$ such that $c = \sum_{i=1}^{K} X_i$.

$$\mathcal{D}_i = ||\hat{c} - \mathbf{U}^T M_i||^2 \tag{1}$$

The projection of the normal subseries forms a dense cluster which is closer to the center. While an abnormal subseries is forced to be projected far away from the center of a normal cluster $(\hat{c})$ thereby having a higher departure score. If the departure score crosses certain threshold $\theta_p$, i.e., if $||\hat{c} - \mathbf{U}^T v_i||^2 > \theta_p$, an attack alarm is triggered.

To compute $\theta_p$, PASAD computes departure scores on training measurements and few extended measurements collected during normal operation. The extended measurements are called validation dataset. PASAD sets $\theta_p = \max_{\forall i}(\mathcal{D}_i)$. As a result, PASAD forms an $\mathbf{n}$-spherical decision boundary (an $n$-sphere is a generalized form

of a sphere in the n-dimensions) in $R$-dimensional signal subspace. The radius of the $n$-sphere is $\sqrt{\theta_p}$ which is the distance of the farthest normal point from center ($\hat{c}$) in the signal subspace.

PASAD is a lightweight IDS suitable for deploying on limited hardware resources. PASAD's most computationally intensive step is to project the $L$-dimensional vector into an $R$-dimensional signal space which is an $R \times L$ dimensional matrix to $L$ dimensional vector multiplication. As a result, the computational complexity of the PASAD is $O(RL)$.

## Attack model

In this section, we discuss the attack model that encompasses the motivation for developing EPASAD along with necessary definitions.

**Definition 1** (*Normal cluster*) Set of normal points (column vectors of **X** in Eq. 2) in signal subspace collected by projecting the measurements when there was no attack (also referred as normal measurements).

**Definition 2** (*Decision boundary*) A non-linear function $f(x)$ encloses the normal cluster and separates the projection of the measurements captured under attack (also referred to as attack measurements) and normal operations.

**Definition 3** (*Tightness of decision boundary*) Let $x_1$ and $x_2$ be two points on a decision boundary $f(x)$, points $x_1'$ and $x_2'$ be the nearest (shortest Euclidean distance) points of the normal cluster from $x_1$ and $x_2$, respectively. The distance between $x_1$ and $x_1'$ be $\delta_f(x_1) = ||x_1 - x_1'||$ defined as tightness of the decision boundary $f(x)$ at $x_1$, similarly for $x_2$. If $\delta_f(x_1) < \delta_f(x_2)$, then the decision boundary $f(x)$ is tighter at $x_1$ in comparison to $x_2$. In other words, $f(x)$ is loose at $x_2$ than $x_1$.

**Definition 4** (*Uniformly tight decision boundary*) Let $f(x)$ and $g(y)$ be the two decision boundaries, and if $|max(\delta_f(x)) - min(\delta_f(x))| < |max(\delta_g(y)) - min(\delta_g(y))|$ , then we call $f(x)$ is more uniformly tight decision boundary than $g(y)$.

### Direct damage attack (DDA)

A DDA is a conventional attacking approach where an attacker does not hide their malicious activities in the physical process. A DDA attacker's goal is to damage the devices and eventually interrupt the process. Here, the attacker tries to accomplish his harmful goals before being detected and make CI operate abnormally. These attacks are trivial to be detected, but any delay in their detection causes severe consequences for a CI. An efficient IDS aims to detect abnormal behavior induced by such attacks at the initial stages to save CIs from lasting damage.

### Stealthy attack (SA)

In Feng et al. (2017), the authors argued that in a noisy environment, a strategic attacker benefits from inflicting a substantial perturbation on the system state. The attack escapes the detection by failure and anomaly detectors as they do not consider noise. Strategic attackers' goal is to cause slow damaging perturbations in the physical process while being undetected for an extended period. Such attacks are likely to produce a cascading effect due to the interaction of control loops, eventually causing the control system to fail. Sometimes a strategic attacker may mask their attack so that the reflected anomaly in physical process variables remains within the noise level; the noise can be manufactured intentionally by the attacker or naturally by the system. Attacks that hide their manipulation within noise margin are known as SAs.

### Micro stealthy attack (MSA)

There have been several attack incidents where attackers compromised CIs by either installing malware, misusing the resources, making user compromise, performing Denial-of-Service (DoS) attacks, making root compromise, and performing social engineering attacks (Kovacevic et al. 2015). An attacker's abnormal activities cause structural changes in the physical process. As attackers aim to cause maximum damage without being detected, a smart attacker hides the abnormalities by controlling the manipulations. There are several other SAs such as those that are model-based advanced SAs. In these types of SAs attackers use control-theoretical knowledge. Some of these SAs are zero-dynamics attacks (Teixeira et al. 2012), poly dynamics attacks (Jeon and Eun 2019), false data injection attacks (Liang et al. 2016), and covert attacks (Smith 2015). Such attacks do not make significant structural changes in the sensor's measurements and are difficult to detect. In this paper, we do not focus on these types of attacks or conduct such attacks. We rather focus on detection of SAs where the sensor measurements are manipulated to cause even minute structural changes in the normal behavior.

In Aoudi et al. (2018), the authors present PASAD that detects such structural changes. However, PASAD has drawbacks. An attacker can evade PASAD by controlling the structural changes. Since PASAD envelops the $R$

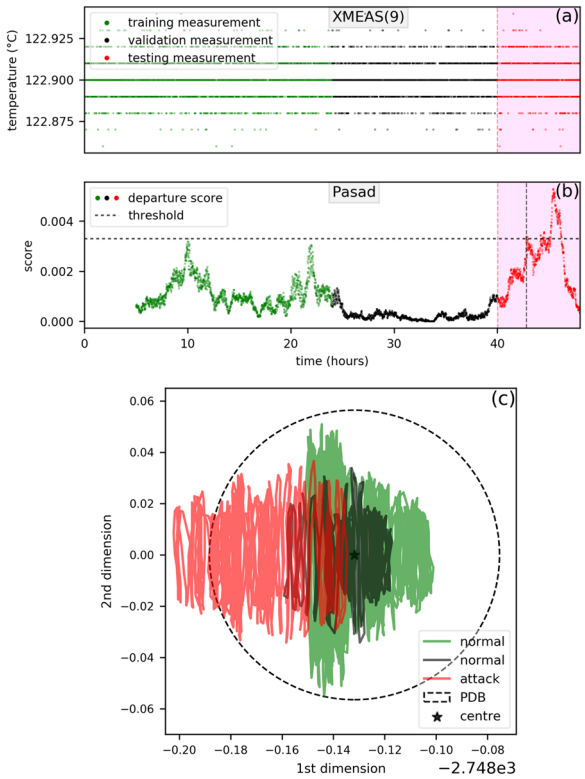


**Fig. 1** We demonstrate a stealthy attack scenario on a reactor's temperature sensor (XMEAS(9)). Here, PASAD framework is delayed in detecting the attack because of the projection of attacked measurements towards the loose side of decision boundary. Subfigure **a** shows the sensor-generated measurements. The green and black measurements are the normal measurements used for training and validation, respectively, while the red measurements are captured under a stealthy attack (SA3). Subfigure **b** represents the departure score of corresponding measurements generated by PASAD frameworks. Subfigure **c** demonstrates the projections of each normal and attack measurement on the signal subspace (we consider a 2-dimensional signal subspace for better visualization) and the PASAD's Decision Boundary (PDB)

-dimensional signal subspace in an n-spherical decision boundary, one side is tight enough while the remaining are loose. There is a high probability that an attack-induced abnormal subseries get projected toward the loose side, or an attacker targets the abnormal projection towards the loosest side to hide the maximum abnormal manipulations. The projection towards the loose side causes serious issues such as delay in detecting the SAs, DDAs, and inability to detected some low-intensity attacks. We refer to such low-intensity SAs as Micro Stealthy attack.

In Fig. 1, we demonstrate the problem caused by a non-uniform loose decision boundary. Figure 1a shows a time series of the reactor's temperature captured by the sensor XMEAS(9) of TE-process, initially under normal (green and black measurements) operation and ended with a SAs (red measurements) operation. We use the measurements under normal operation (green measurements) to determine **P**. The other points under normal conditions (black measurements) determines the decision boundary. Finally, we test the model using the measurement (red measurements) captured under attack. Figure 1b demonstrates the departure score of each sensor measurements computed by PASAD framework.

We further demonstrate the projections of each normal and attack measurement on a 2-dimensional signal subspace (cf. Fig. 1c) for better visualization. In this 2-dimensional signal subspace, the red points (attack subsequence projections) are projected far enough away from the green point's cluster. Since the abnormal projections are towards the loose side, it takes a long time to cross the spherical decision boundary of PASAD, causing a delay in detecting the SA. Thus, a question arises: *What if a strategic attacker slightly reduces the SA's impact and attempts an MSA, never to cross the decision boundary?* PASAD will not detect the MSA attack that silently damages the CI and wastes valuable resources. We demonstrate such MSA attack scenario using Fig. 2. Figure 2a represents measurements generated by sensor XMEAS(21) (represents reactor's cooling water outlet temperature) captured under an MSA scenario (cf. "The Tennessee-Eastman process dataset (TE-dataset)" section–MSA1). Here, the attacker manipulates the purge valve (XMV6) slightly higher than normal with the objective of wasting the reactor's gases. Figure 2c shows that the attack-induced manipulated measurements are projected far enough from the normal cluster. Since the projections are toward the loose side and the impact of the attack is not that high to cross the decision boundary, the departure score of PASAD has never crossed the threshold (cf. Fig. 2b) and fails to detect the attacks reflected in XMEAS(21). Thus, we introduce EPASAD with a motivation to quickly detect the MSA, SA, and DDA.

## Proposed framework: EPASAD

EPASAD is a process-level, univariate, and anomaly-based IDS framework that monitors ICS process activity in real-time to determine whether the system is under normal or abnormal operation. Due to SSA's noise cancellation property, EPASAD works even in a noisy environment.

EPASAD collects the set of normal subseries on the signal subspace and envelops it within an efficient decision boundary. The subseries captured under normal

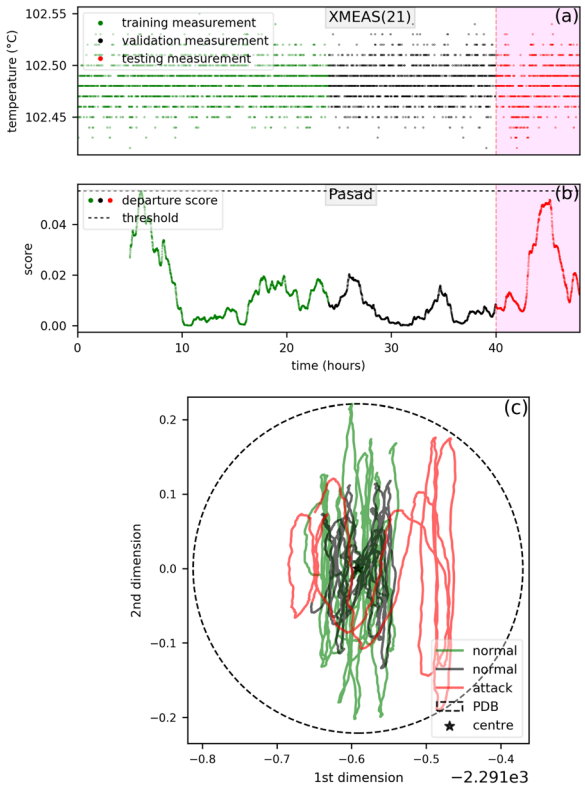


**Fig. 2** We demonstrate an MSA scenario where PASAD framework fails to detect the attack because of the attack's projection towards the loose side. Subfigure **a** shows measurements generated by the reactor's cooling water outlet temperature sensor (XMEAS(21). The red measurements are captured under a micro-stealthy attack (MSA1). Note that all other aspects and subfigures have same definition as Fig. 1

operation follow certain oscillation and trend structures, projecting a set of normal subseries that forms a dense cluster of normal points. While an abnormal subsequence that has some structural manipulations get projected far from the normal cluster. An attack alarm is triggered if the projection surpasses the decision boundary.

EPASAD uses the normal cluster to learn a uniformly tight and computationally efficient decision boundary. Many nonlinear functions such as convex/non-convex hull, skewed ellipsoid, higher-order nonlinear functions can envelop the signal subspace. Nonetheless, we use a specific ellipsoid function to parallel the standard axis of signal space to avoid any increase in online testing computation cost while ensuring a uniformly tight decision boundary for every dimension. We demonstrate EPASAD using Fig. 3. Figure 3a represents the same attack scenario demonstrated in the Fig. 1. Figure 3d shows an elliptical curve enveloping the 2-dimensional signal space within a minimum area. It brings the loose side of the decision boundary closer to the normal cluster, making each dimension uniformly tight. The elliptic decision boundary easily separates the abnormal red points that the spherical decision boundary misses. Hence EPASAD creates a challenging decision boundary for an attacker but is simpler to deploy. It does not give any redundant normal subspace where attacker can hide his abnormal activities.

## Training of EPASAD framework

Consider a real-valued univariate time series $\mathcal{T} = [m_1, m_2, \ldots, m_N, \ldots, m_{N'}, m_{N'+1}, \ldots]$. The subseries from $m_1$ to $m_N$ is used to determine $\mathbf{P} = \mathbf{U}^T$ while from $m_{N+1}$ to $m_{N'}$ as validation dataset. Before proceeding with the section, we list our assumptions.

### *Assumptions*

There are three basic assumptions to develop the EPASAD framework: (i) the dataset used for training EPASAD can be noisy but cannot be anomalous. An anomalous pattern in training data can cause a data poisoning attack. (ii) EPASAD is trained in an offline fashion, which needs all the training and validation datasets of length $N'$ to be available during training. (iii) EPASAD prepares input features with the help of recent measurements that require an uninterrupted sequence of measurement.

### *Step 1: generate normal cluster*

We collect the normal cluster by projecting the normal lagged vectors into the noise-free signal subspace. To determine $\mathbf{P} = \mathbf{U}^T$, EPASAD is trained over $\mathcal{T}[1:N]$ by utilizing the SSA and PASAD. The projection matrix projects an $L$-dimensional lagged vector from real space to an $R$-dimensional ($R \leq L$) signal subspace. The projection matrix is trained over the series has a possibility of over-fitting the training data. Hence, we extend the normal training subseries with the validation datasets extending from $N$ to $N'$ ($N' > N$), i.e., ($\mathcal{T}$[1:$N'$]). Thus, the trajectory matrix $\mathbf{M}$ for the extended validation subseries is of size $L \times K'$, where $K' = N' - L + 1$ and each column vectors of $\mathbf{M}$ are projected to a signal matrix $\mathbf{X}$ of size $R \times K'$. The $i$th column vector is projected as $\mathbf{X}_i = \mathbf{U}^T M_i$. Hence, using Eq. 2 we project the entire $L$−dimensional matrix $\mathbf{M}$ to an $R$−dimensional signal matrix $\mathbf{X}$.

$$\mathbf{X} = \mathbf{U}^T \mathbf{M} \tag{2}$$

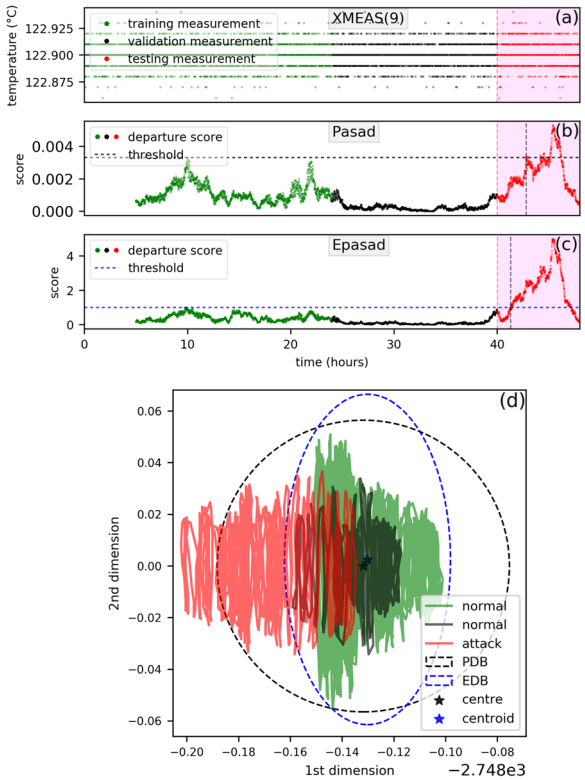


**Fig. 3** We demonstrate a stealthy attack scenario and its detection. Our proposed framework EPASAD is able to detect the attack more quickly than the baseline method PASAD. Subfigure **a** shows a sensor-generated measurements (by XMEAS(9) sensor, represents reactor's temperature). The green and black measurements are normal measurements used for training and validation, and the red measurements are captured under a stealthy attack (SA3). Subfigures **b** and **c** represent the departure score of corresponding measurements generated by PASAD and EPASAD frameworks. Subfigure **d** demonstrates the projections of each normal and attack measurement on the signal subspace (we consider a 2-dimensional signal subspace for better visualization) and the decision boundaries of both, i.e., PASAD's decision boundary (PDB) and EPASAD's decision boundary (EDB)

#### *Step 2: finding centroid*

We estimate the centroid $\hat{c} \in \mathbb{R}^R$ of the ellipsoid decision boundary using Eq. 3. Here, the elements of vector $min(\mathbf{X})$ are minimum elements of the corresponding dimension of $\mathbf{X}$ similarly, $max(\mathbf{X})$ are maximum elements. The mean of the cluster of a skewed sample distribution shift towards the dense side. Considering the mean as the centroid of the ellipsoid makes the decision boundary envelop the sparse side tightly and the opposite side loosely. Therefore, rather than choosing the projection of the mean of cluster to determine centroid as in PASAD, we determine the mid-point of the range of each dimension of $\mathbf{X}$. Further, we make centroid invariant signal subspace by using Eq. 4 where $\mathcal{C}(x)$ is a centroid invariant element-wise squared vector. The centroid invariant signal subspace standardizes the ellipsoid decision boundary centered around zero-vector for every sensor.

$$\hat{c} = \frac{min(\mathbf{X}) + max(\mathbf{X})}{2} \tag{3}$$

$$\mathcal{C}(x) = (x - \hat{c})^2 \tag{4}$$

#### *Step 3: learning ellipsoid decision boundary*

We determine the ellipsoid decision boundary that envelops the normal cluster in signal subspace $\mathbf{X}$. We consider a hypothesis function $f(x)$ for a variable vector $x \in \mathbb{R}^R$ to learn the decision boundary (cf. Eq. 5, here $w$ is a weight vector). When we express the hypotheses function $f(x)$ in the form of a standard ellipsoid function, the $\sqrt{w_i}$ describes the length of $i$th axis of the ellipsoid.

$$\begin{aligned} f(x) &= w^T \mathcal{C}(x) \\ &= \frac{(x_1 - \hat{c}_1)^2}{(w_1^{-0.5})^2} + \frac{(x_2 - \hat{c}_2)^2}{(w_2^{-0.5})^2} + \cdots + \frac{(x_r - \hat{c}_r)^2}{(w_r^{-0.5})^2} \end{aligned} \tag{5}$$

Our aim is to minimize the generalized $n$-dimensional volume to get minimum void space inside the decision boundary. Thus, we minimize the length of each ellipsoid axis such that all points of the normal cluster remain inside $f(x)$. Since the product of axis length is proportional to the ellipsoid volume, Eq. 6 is our objective function for learning the hypothesis function $f(x)$. Solving the objective function returns an optimal weight vector $\hat{w}$ that minimizes the product of the length of each axis ($\prod(w)^{-0.5}$). There are two hard constraints associated with the objective function 6: *(i)* $w^T\mathcal{C}(x) \leq 1$, forces each point to remain inside $f(x)$, and *(ii)* $w > 0$ assures an ellipsoid's real-valued axis length. We train the objective function over the column vectors of signal matrix $\mathbf{X}$ that gives an optimal weight vector $\hat{w}$ to get an optimal decision boundary.

$$\hat{w} = \arg\min_{w} \left( \prod (w)^{-0.5} \right) | w^T \mathcal{C}(x) \leq 1, \ \forall x \in X \ \& \ w > 0 \tag{6}$$

#### *Step 4: set threshold*

Since we train the objective function to minimize length of each axis of decision boundary using a hard constraint $w^T\mathcal{C}(x) \leq 1$, the value of $f(x)$ at a threshold $\theta_e = 1$ is a decision boundary. The function $f(x)$ forms the tightest enveloping function $f(x)$, which does not consider any margin of error. However, a normal measurement can slightly deviate

from the normal cluster causing false alarms. Thus, we add a margin of error, $\epsilon$, also called slack-value in the threshold, $\theta_e = 1 + \epsilon$, to control the false alarms.

#### Testing EPASAD framework

The EPASAD framework is deployed over SCADA to test each live streaming measurement in an online fashion. If $m_t$ is a measurement generated at timestamp $t$ and received by the SCADA, EPASAD prepares an $L$ length lagged vector $M_t$ using previous $L-1$ measurements; $M_t = [m_{t-L}, m_{t-L+1}, \ldots, m_t]^T$. The real-space lagged vector $M_t \in R^L$ are projected onto the $R$-dimensional signal subspace; $X_t = \mathbf{U}^T M_t$. For the most recent test measurement $m_t$, EPASAD computes a $\mathcal{D}_t = f(X_t)$. The $\mathcal{D}_t$ describes the confidence, regardless of whether the measurement is classified as an attack or normal. A smaller $\mathcal{D}_t$ indicates greater confidence of a measurement to be normal, while a higher $\mathcal{D}_t$ indicates greater confidence of an attack. A test measurement is classified as normal up to a tolerable value of the departure score threshold $\theta_e$. If $\mathcal{D}_t \geq \theta_e$, then EPASAD raises an attack alarm. This process completes the online testing step for a single measurement. The same procedure is repeated for the subsequent measurement generated at time $t+1$, and so on. The Algorithm 1 depicts the pseudo-code of the EPASAD framework's online testing phase.

**Algorithm 1:** EPASAD's online testing

```
input : Lag parameter L, Dimensionality of signal
        subspace R
output: An alarm when attack is detected
Data: A test sequence T
determine U            // Using SSA during training
determine c            // The centroid of ellipsoid
determine w            // By Equation 6
θ_e ← 1 + ε            //Set threshold
while (1) do //Online testing the measurements streams
    m ← [m_{t-L}, m_{t-L+1}, ··· , m_t]  //Test subsequence
    x ← U^T m //Project m to R dimen. signal subspace
    y ← (x − c)^2
    D_t ← w^T y          // Departure score for m_t
    if D_t > θ_e then
        Raise an attack alarm
    end
end
```

#### Computation cost

An IDS is deployed for the long term to secure the real-time streaming measurements from sensors. A sensor associated with ICS regularly sends measurements to the IDS; there may be a small-time difference between the streaming measurements. The IDS deployment must be efficient enough to generate the decision before proceeding to the subsequent measurement. Hence, online testing is crucial for low-cost hardware deployment. On the other hand, training is typically one time task accomplished in an offline fashion.

The main computation cost of EPASAD is the computing the departure score. The departure score evaluates a matrix to vector multiplication $x \leftarrow \mathbf{U}^T m$, it multiplies a $R \times L$ matrix to an $L$—dimensional vector requires $\mathcal{O}(RL)$ computing cost. Then, $y \leftarrow (x-c)^2$ is an element-wise operation of two $R$-dimensional vectors with $\mathcal{O}(R)$ complexity. The final computation steps $D \leftarrow w^T y$ requires a dot product of two $R$—dimensional vectors, $\mathcal{O}(R)$. Hence, the overall computation cost of EPASAD is $\mathcal{O}(RL + R)$, which is equivalent to the computation cost of PASAD. Usually, only a few leading eigenvectors retain the majority of the signal information. Therefore, $R << L$ is the average case of the computation cost. In the average case, the time complexity for online detection of EPASAD is linear in $L$, i.e., $O(L)$. The online deployment of EPASAD needs to store a projection matrix $\mathbf{U}^T$, centroid $c$, weight vector $w$, Which is require space to keep $RL$, $R$, and $R$ real numbers, respectively. Hence the space complexity of EPASAD is $\mathcal{O}(RL)$. Compare to PASAD, EPASAD needs to store an addition $R$-length weight vector $w$ which does not contribute much to space complexity. Hence both PASAD and EPASAD have the same space complexity of $\mathcal{O}(RL)$.

## Validation datasets

We validate our proposed methodology using multiple attacks scenarios present in the two datasets listed below:

#### The Tennessee-Eastman process dataset (TE-dataset)

The TE-dataset is generated using an industrial chemical process simulation model proposed in 1993 (Downs and Vogel 1993). The TE simulation framework mimics the process in a real-world chemical plant. The TE-process serves as a more realistic and safe environment for experimentation, transcending its original objective and becoming a popular choice among ICS security researchers (Aoudi et al. 2018; Zhu et al. 2017; Gao and Hou 2016). The TE process has 12 cross-correlated Manipulated Variables (XMVs) and 41 cross-correlated MEAsured variableS (XMEAS). XMEAS(i) represents measured values by the $i$th sensors, and XMV(i) represents the $i$th variable, which can be manipulated to collect the measured values. In Aoudi et al. (2018), the authors considered five attack scenarios to validate their method: three SAs and two DDAs. We consider

two additional attack scenarios representing MSA and generate the TE-dataset by performing the following attacks.

#### *Micro-stealthy attack (MSA)*

We consider two MSA attack scenarios to validate EPASAD. These include:

- *MSA1* We simulate this attack by manipulating the process variable of a purge valve (XMV(6)). The XMV(6) restrict the reactor gas in the reactor tank from escaping into the atmosphere. Unnecessarily opening the valve more than a certain level causes low pressure in the reactor; Thereby causing the process to halt. Also, it causes unnecessary wastage of valuable gasses. In this scenario, we open the valve by 26%, which is enough to degrade the system and waste the reactor gases but not that high to interrupt the process.
- *MSA2* We simulate this attack by manipulating the speed of an agitator (XMV(12)). The agitator ensures a well-mixed reactor, which impacts the heat transfer coefficients in the reactor. The maximum speed of the agitator should be 100% to maximize the cooling capacity of the reactor coolant, and ideally, it is suggested to be 50% (Downs and Vogel 1993). Hence, reducing the agitator speed below 50% can increase the reactor's temperature, causing damages to the system. In this attack scenario, we consider the 38% speed of the agitator which is slow enough to reduce the coolant capacity and increase the reactor's temperature.

#### *Stealthy attack (SA)*

We consider three SA scenarios:

- *SA1* We simulate this attack by manipulating the Stripper steam valve XMV(9). This valve controls the steam input to the stripping column. In this attack, we open the valve at 40% compared to completely open.
- *SA2* We simulate this attack using the MSA1 attack scenario with a higher impact. In this attack scenario, we open the purge valve by 28%, 2% more than in MSA1.
- *SA3* We simulate this attack by tampering with the sensor XMEAS(10) to zero. The zero measurements of XMEAS(10) represent that purge valve XMV(6) is closed. For the counteraction, the controller would unnecessarily open the purge valve.

#### *Direct damage attack (DDA)*

We consider two DDA scenarios:

- *DDA1* We simulate this attack by manipulating the process variable XMV(10) of a valve that controls cooling water flow to the reactor to prevent its temperature and pressure reach at a dangerous level. In this scenario, we open the valve to 35.9%, which is lower than usual (41.106%). Consequently, it increases the reactor's pressure and temperature and stops the process from reaching the maximum predefined limit.
- *DDA2* We tamper the reactor pressure sensor XMEAS(7) to zero. The controller takes action to perform more chemical reactions to maintain the reactor pressure. The abnormal increase in the pressure can damage the reactor, eventually stopping the process.

Each attack scenario of TE-dataset consists of measurements of 41 sensor as a time series. The dataset is collected for 48 hours, with the initial 40 hours under normal operation, and the remaining last 8 hours are during an active attack. The measurements are generated periodically such that it takes one hour to generate 100 measurements.

#### C-town dataset

The C-town network dataset (Taormina et al. 2018) is generated by simulating Epanet CPA (Taormina et al. 2017). The network consists of 43 sensors and generates a measurement after every hour periodically. The dataset contains 14 distinct attacks launched in a different time window throughout nine months. The dataset contains three subdatasets, each of which consists of 43 process variables:

- *Subdataset 1* It contains normal measurements during a period of one year.
- *Subdataset 2* It contains seven attacks along with normal operations during a period of six months.
- *Subdataset 3* It also contains seven attacks (but different) along with normal operations during a period of three months.

Each subdataset, as mentioned above, is collected for the same sensor network. We combine subdataset 2 and 3 and call it subdataset 4 to evaluate EPASAD on the 14 attack scenarios captured during the nine-month-long period. The details of each attack scenario are provided in the paper (Taormina et al. 2018).

## Experiments and results

In this section, we validate our proposed method using above mentioned datasets and provide parameter values selected for the experiments.

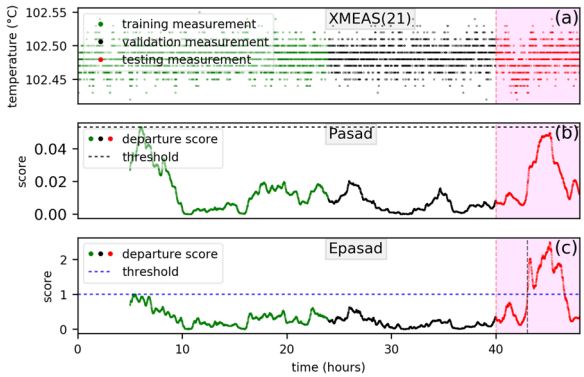


**Fig. 4** We show the comparison of PASAD and EPASAD over sensor XMEAS(21) of TE-dataset. The attack measurements are collected during a micro-stealthy attack (MSA1) operation. EPASAD is able to detect the MSA, which PASAD fails to detect

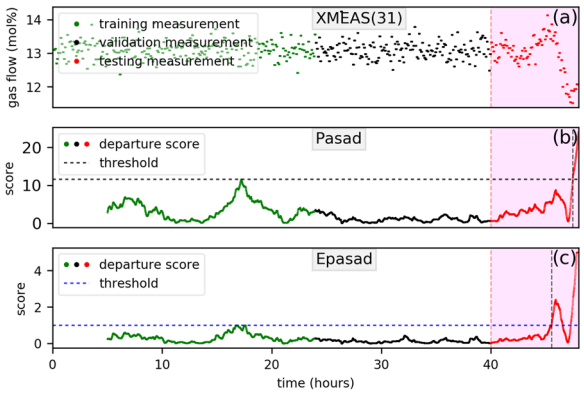


**Fig. 5** We show the comparison of PASAD and EPASAD over purge gas analysis stream sensor XMEAS(31) of TE-dataset. The attack measurements are collected during a Direct damage attack (DDA1) operation. EPASAD is able to detect the DDA more quickly

**Experiment on TE-dataset**

In this experiment, we study how quickly we can detect SAs, MSAs, and DDAs. This experiment is carried out using comparable datasets and parameters for training, validation, and testing to make a fair comparison with the baseline method PASAD. Hence, we consider the normal subseries of the first 2400 (green) measurements to get the projection matrix $\mathbf{U}^T$ and then use the remaining 1600 (black) normal measurements as the validation dataset along with the training set to obtain the EPASAD decision boundary. We then apply the entire time series to EPASAD to do online testing. Figures 3, 4, and 5 demonstrate the effectiveness of EPASAD towards detecting different attack scenarios in the TE process and comparing it with the baseline line method. Figures 3a, 4a, and 5a represent the time series of sensor measurements. Figures 3b, 4b, and 5b represent the corresponding departure score by applying the baseline method PASAD. Figure 3c, 4c, and 5c represents the departure score by applying our proposing method EPASAD. Similar to Aoudi et al. (2018), we also set the threshold at maximum departure score of the normal measurements hence there was no false-alarm in the TE-dataset scenarios. Therefore, all the evaluation of this dataset is represented in term of recall only.

Figure 3 shows sensor operating under SA scenario. The part of the subseries that has been captured under SA appears to be normal. Such anomalous series when projected on the signal subspace are significantly far from the normal cluster. PASAD's departure score takes a long time to be more than $\theta_p$, causing a delay in detecting the attack. Moreover, the departure score raising alarm returns to normal after a short period, which an administrator may think of as a false alarm. On the other hand, EPASAD detects the attack shortly after it begins and raises the alarm for an extended time. Hence, EPASAD is more effective at detecting SAs quickly. Further, we evaluate EPASAD on each process variable of SA scenarios SA1, SA2, and SA3. Our results (cf. Fig. 6) show a significant improvement in all the attack scenarios. EPASAD improves the average recall of all three SAs from 50.3% to 54.2% compared to the baseline benchmark.

We demonstrate our method on a process variable which is captured under MSA (cf. Fig. 4). The results show that the departure score of PASAD is always less than the $\theta_p$ during the attack. Hence, it could not detect MSA. On the other hand, EPASAD computes a significant departure score which is more than the $\theta_e$ for a lengthy period. Hence, EPASAD is able to detect even the MSA. We tested EPASAD on every process variable in the MSA1 and MSA2 datasets. The results (cf. Fig. 6) show significant improvement with the average recall increasing from 7.5% to 17.3%.

We evaluate our method on a process variable of the DDA1 attack scenario (cf. Fig. 5). In this scenario, the measurements during the attack operation are initially close to normal and then suddenly become abnormal, even beyond the normal range (the lower and upper limit of measurements generated by a sensor). The baseline method PASAD could not recognize the initial symptoms. It detects the attack when the attack induced-measurements reach beyond the normal range. On the other hand, EPASAD detects such attacks at early stages, shows a significant gain over the baseline method. Hence, EPASAD can quickly detect the DDAs. Figure 6 shows the average performance of EPASAD on each process variable of the DDA1 and DDA2 attack scenario. Here, EPASAD improves recall score from 46.2% to 51.0%.

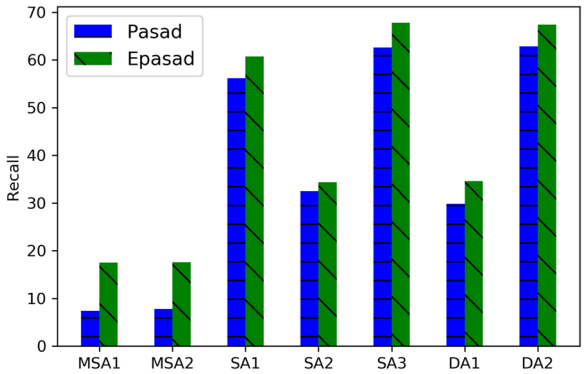


**Fig. 6** We compare the accuracy of PASAD and EPASAD over the seven different attack scenarios of the TE-dataset in terms of recall. EPASAD achieves a significant improvement in each attack scenario

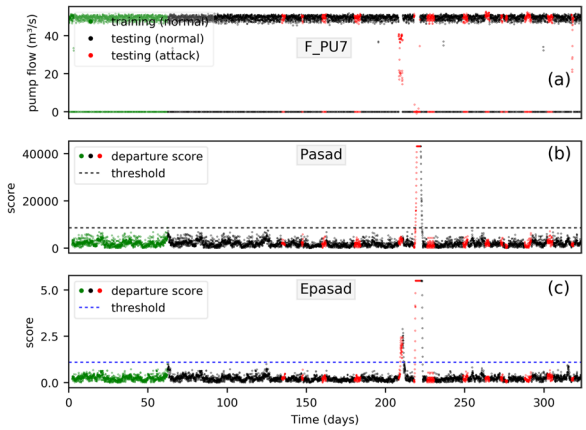


**Fig. 7** Comparison of PASAD and EPASAD over the pump-flow sensor (F_PU7) of C-town dataset, collected during 14 different attacks where PASAD fails to detect the abnormality induced during the 5th attack, and EPASAD is able to detect it. The green measurements are normal measurements used for training. The black and red measurements are normal, and attack measurements are used for testing. Note that the order of each subfigure has the same definition as Figs. 4 and 5

### Experiment on C-town dataset

In a realistic scenarios, attacks are launched for a limited duration, and then the system resumes normal operation. The 14 attacks in this experiment are launched for a limited time before the system resumes normal operation. This is recurrent and done over a period of 9 months. Figure 7 demonstrates EPASAD on a process variable of the C-town dataset. We train EPASAD over a subseries of length 1500 (green measurement) captured under normal operation to get the projection matrix. Then, include 1500 normal measurements (black measurements next to the green ones) as validation dataset to determine the decision boundary. Once the training phase is complete, we test the entire subseries using the online testing algorithm 1. Figure 7c indicates EPASAD's strengths in detecting the structural changes caused by the 5th and 6th attacks reflected in the *FPU7* sensor and then return to the normal state.

In Tables 2 and 3, we evaluate the experiment at the entire infrastructure level by aggregating the nature of alarms in every process variable. If the IDS triggers an alarm in any processes during an attack, we consider the attack to be detected. We consider a false alarm if it is triggered in any sensor during the normal operation. Table 3 evaluates each attack using two attributes, time (in hours) and count. The time field represents how long an attack has been active without causing an alarm to be raised. In other words, it is the time taken by IDS to raise the first alarm. The count field represents the number of process variables involved in the alarm's triggering. It is very unusual to raise false alarms in multiple sensors at a time, a higher number of counts sensors producing attack alarm increases the confidence of positive alarm. Table 2 evaluates the overall accuracy in terms of true alarm rate (recall), precision, F1-score, and false alarm rate.

This experiment tests the long duration when measurements are captured under mostly normal operation and sometimes under various attacks. Hence, there is a possibility that an IDS in this experiment generates a large number of false alarms. The results in Table 2 show a significant improvement by EPASAD in the precision, recall (true alarm rate), F1-score, and a low false alarm rate as compared to PASAD. In addition to PASAD, we evaluate the other benchmark methods (Hadžiosmanović et al. 2014; Aoudi and Almgren 2020; Dutta et al. 2021; Goh et al. 2017; Taormina and Galelli 2018) under the same experimental setup. The comparative analysis is shown in Table 2. Here, EPASAD is identified as the best-performing method, while PASAD performs best among the other baseline methods.

In addition to the overall performance, we analyze the detection of all 14 attacks in Table 3. As EPASAD is an extension of PASAD and performs the best among the baseline methods, we do an additional comparative analysis of PASAD and EPASAD for every 14 attack scenarios. We analyze the time taken to detect an attack and the count of the number of sensors engaged in triggering an alarm. EPASAD has a significant gain in detecting the two attacks (9th, and 12th) over PASAD, and EPASAD even detects the two missing (2nd and 8th) attacks. EPASAD generates a valid alarm in more number of sensors that increase the alarm's confidence. Hence, EPASAD can quickly and confidently raise the alarm for detecting an attack. EPASAD slightly under-performs in

**Table 2** The average performance and comparison (in percentage) of EPASAD, PASAD, and other baseline frameworks on the C-town dataset

| Methods | Precision | Recall | F1-score | False Alarm |
|---|---|---|---|---|
| EPASAD | 71.36 | 64.29 | 67.64 | 3.70 |
| PASAD (Aoudi et al. 2018) | 64.36 | 54.84 | 59.22 | 4.36 |
| AR (Hadžiosmanović et al. 2014) | 32.37 | 53.99 | 40.47 | 3.96 |
| MPASAD (Aoudi and Almgren 2020) | 57.17 | 43.86 | 49.64 | 10.50 |
| RPCA (Dutta et al. 2021) | 24.36 | 26.01 | 25.16 | 9.94 |
| LSTM (Goh et al. 2017) | 54.39 | 61.36 | 57.67 | 4.92 |
| AE (Taormina and Galelli 2018) | 54.95 | 58.05 | 56.46 | 5.70 |

**Table 3** Performance and comparison of PASAD and EPASAD framework for all 14 attacks present in the C-town dataset

| Attack | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Time (EPASAD) | 0 | 10 | 3 | 16 | 12 | 16 | 32 | 18 | 0 | 0 | 10 | 18 | 4 | 10 |
| Time (PASAD) | 0 | × | 4 | 17 | 10 | 14 | 26 | × | 22 | 0 | 10 | 52 | 4 | 17 |
| Count (EPASAD) | 6 | 2 | 10 | 6 | 6 | 5 | 5 | 3 | 8 | 7 | 9 | 6 | 4 | 2 |
| Count (PASAD) | 6 | 0 | 9 | 6 | 3 | 5 | 5 | 0 | 4 | 6 | 11 | 3 | 3 | 2 |

The comparisons are based on the time (in hours) taken to detect an attack and the number of sensors that trigger the alarms. Here '×' represents an attack not detected

three scenarios (cf. 5th, 6th, and 7th attack scenario in Table 3) of the C-town dataset. In PASAD, if the projection is on the tight dimension, it performs slightly better. EPASAD slightly loosens each dimension by adding a small value "slack" to the threshold. Thus, if the projection is in the tight dimension, PASAD might be better. But in general, keeping the attack such that all dimensions are tight is hard for the attacker to find the loosest side as they would now need to identify radii in each dimension.

### Parameter selection

In this section, we discuss the parameters and their choices that help us in implementing the above experiments. We use the same datasets and parameters to experiment with PASAD and EPASAD to make a fair comparison. There are two main parameters that are required in the training phase: lag $L$ and dimensionality of signal subspace $R$. The lack of generalization of parameters in the baseline paper encourages us to choose the best performing parameter for PASAD. We run PASAD over various lag values, from 100 to 1000 in the increments of 100 for the TE dataset and 20 to 200 in increments of 10 for the C-town dataset to find the best lag value. We find the best performing lag parameter, $L$=500 for TE-dataset and $L$=50 for the C-town dataset. A smaller value of the lag parameter for the C-town dataset yield the best results because the time between two consecutive measurement is one hour, while the TE-dataset generates 100 measurements in one hour. Hence, a subsequence of length 50 itself covers the subsequence of more than two days. The dimensionality of signal space $R$=3 is found to be best performing. Once the training is finished, we set a threshold $\theta_e$ to classify the departure of measurement between attack and normal. The experiment "Experiment on TE-dataset" of the TE dataset uses entire normal subseries for training and validation, which ensure no false alarm with a minimum threshold with slack-value $\epsilon = 0$. In the experiment "Experiment on C-town dataset", when we set $\theta_p$ to the maximum of validation subseries without adding any slack-value, we find that PASAD fails to detect two attacks (2nd and 8th). Adding a slack-value could fail to detect more attacks and decreases the alarm. On the other hand, EPASAD is tighter in each dimension has a higher chance of raising a false alarm. Hence, we add a slack-value $\epsilon = 0.1$ in $\theta_e$ to ensure a lesser false alarm rate.

## Related works

In this section, we discuss earlier IDSs in the industrial control system. In Aoudi et al. (2018), the authors published a method to detect attacks in ICS at a process variable label named PASAD. PASAD is a univariate departure-based process-level detection method that can detect even a SA on control systems by identifying an abnormal sequence. There are two other popular process level detection methods: Linear Dynamic State-space (LDS) by Shoukry et al. (2015) and the Auto-Regressive

(AR) methods (Hadžiosmanović et al. 2014) (which we describe later in the section). A comprehensive survey of these methods is presented in Urbina et al. (2016).

Along with the univariate process-level detectors, there are other popular multi-process-level detectors methods. In Guan et al. (2003), the authors used the K-Means clustering method along with the algorithms discussed in Hansen and Mladenović (2001) and named it Y-Mean clustering method for network intrusion detection. This method is tested on the KDD99 dataset. In Hu et al. (2008), the authors applied the AdaBoost algorithm on the KDD99 dataset and achieved better accuracy with fewer false alarms. In Nader et al. (2014), the authors used one-class SVM with kernel PCA to detect attacks in the Gas Pipeline testbed and water treatment plant (Lichman et al. 2013). Further, different studies also use reconstruction-based deep learning methods (Feng et al. 2017; Goh et al. 2017; Taormina and Galelli 2018). In Feng et al. (2017), the authors combined the Long Short-Term Memory (LSTM) network with a bloom filter to detect the malicious traffic in the gas pipeline SCADA dataset. In Goh et al. (2017), the authors predicted the next measurement using the LSTM and checked both positive and negative deviation from actual measurement, validating the method on water treatment testbed datasets. Similarly, in Taormina and Galelli (2018), the authors used the AutoEncoder model to reconstruct a measurement, and if it is found a higher deviation from the actual, then trigger an alarm. The method is further improved by using cumulative sum (CUSUM). In Aoudi and Almgren (2020), authors leveraged multivariate SSA and proposed MPASAD, where the main objective was to develop a computationally efficient approach. In Dutta et al. (2021), the authors developed a multivariate IDS using a robust PCA-based dimensionality reduction method.

A process-level IDS is categorized in two categories, the univariate (independent IDS for each sensor variables) (Aoudi et al. 2018; Shoukry et al. 2015; Hadžiosmanović et al. 2014; Aoudi and Almgren 2021) and multivariate (an IDS model takes input from the multiple sensor variables) (Guan et al. 2003; Hansen and Mladenović 2001; Hu et al. 2008; Nader et al. 2014; Feng et al. 2017; Goh et al. 2017; Taormina and Galelli 2018; Aoudi and Almgren 2020; Dutta et al. 2021). In Garcia et al. (2017), the authors developed a PLC rootkit that can corrupt the communication route between sensors and SCADA. An attacker can compromise a few communication channel and manipulate them accordingly to misclassify the structural changes in any other sensors as well. In Erba et al. (2020); Biggio and Roli (2018), the authors used this concept to construct an evasion attack against multivariate detectors (Feng et al. 2017; Goh et al. 2017; Taormina and Galelli 2018). On other hand, an univariate detectors are independent model for each sensor. Manipulating a few sensor measurements cannot evade any other univariate IDS model.

There are four univariate process-level-based detectors methods: LDS methods, AR methods, PASAD and PADS. In Urbina et al. (2016), the authors survey and explain a model that uses the LDS method with a time delay to detect the pH water level using SWaT testbed (Mathur and Tippenhauer 2016). In Cardenas et al. (2011), the authors created several TE process attacks and used LDS together with non-parametric CUSUM statistics. In Shoukry et al. (2015), the authors used the model together with $\chi^2$ anomaly detection technique to extend it for various kinds of sensor variables named it PyCRA. These LDS-based methods are challenging to build. They need a detailed description of process variable that may not always be available (Feng et al. 2017; Kiss et al. 2015). In Hadžiosmanović et al. (2014), the authors leveraged auto-regressive model with Shewhart control limits on time series extracted from the Modbus PLC traffic, evaluated their approach on two water treatment testbed datasets. The result of this method is compared with the PASAD in Aoudi et al. (2018). The authors found that the AR model fails to detect the SAs and delay detecting the DDA; hence PASAD is found more substantial to detect those attacks. In Aoudi and Almgren (2021), the authors present another univariate framework called PADS, which uses departure score of PASAD to classify an alarm in two categories, weak alarm and actionable alarm using two thresholds setting. This framework determines a higher threshold that classifies the alert as an actionable alert. It reduces the frequency of false alarms also recall. Similarly, for weak alert, it increases the false alarm rate as well as recall. Hence, it is difficult to compare the results with this framework. Since EPASAD is improving the departure score of PASAD can improve PADS as well.

In summary, two major categories of process-level IDSs are classified- univariate and multivariate. Multivariate IDSs suffer from the vulnerability of evasion attacks, while the independent nature of univariate makes them secure. We find four univariate detector methods (Aoudi et al. 2018; Shoukry et al. 2015; Hadžiosmanović et al. 2014; Aoudi and Almgren 2021) where PASAD is the most accurate and efficient univariate process-level data-driven method to detect attacks in critical infrastructures, therefore we consider PASAD for baseline comparison. Our proposed method EPASAD improves the performance without hurting its any strengths. The detailed comparison of EPASAD with PASAD and other baseline methods by using two popular benchmark shows that the proposed method EPASAD is more accurate than PASAD, and it detects attacks that PASAD fails to detect.

## Discussion and conclusion

The CIs are vulnerable to cyber-attacks, primarily due to the importance of CIs to the nation and society. In a world full of threats, attackers successfully breach the many tiers of CI security. This research presents a last-layer security solution called EPASAD framework to detect an attack after an attacker has successfully evaded all network security and begun harming the CIs. EPASAD is a univariate, light-weighted, process-level, non-parametric, data-driven, and model-free attack detection framework, that is motivated to detect even tiny structural changes hidden within the noise margin of a process variable. To validate the EPASAD framework, we introduce a MSA scenario, which is extremely difficult to detect by any available methods, but EPASAD efficiently detects it. EPASAD detects quickly every other attack scenario considered for validation and significantly improves the performance of PASAD without any additional computational overhead. We summarize the following six essential strengths of EPASAD based on our experiments on various attack scenarios and available literature:

- *EPASAD quickly detects an attack* EPASAD aims to detect even tiny structural changes in the normal behavior of the sensor and detect even MSA attack at the very initial stages (cf. Fig. 4). Based on the experiments performed, EPASAD improves the performance of detecting the attacks in all attack scenarios, including seven of TE-dataset and fourteen of C-town dataset (cf. "Experiments and results" section ). In a most unlikely scenario, when the signal space is equally distributed across each dimensions, EPASAD can still learn a uniformly tight n-spherical decision boundary. Thus, EPASAD's performance will always be better than PASAD.
- *EPASAD also works under noisy environment* In Mo and Sinopoli (2015), the authors highlighted the critical problem of making the unrealistic assumption that the system model is noiseless. A noisy environment can cause severe problems for a non-robust IDS. An attacker can hide their malicious manipulations within the noise, and the noisy environment causes lots of false alarms. Our proposed method, EPASAD, is based on a well-known robust time series tool called SSA. The SSA is suitable to capture the skeleton of deterministic pattern from a noisy time series that makes EPASAD robust enough to work even in a noisy environment (cf. Chapter 6 of Elsner and Tsonis 2013).
- *EPASAD is realistic to build and deploy* EPASAD is a non-parametric and purely data-driven framework that does not need prior knowledge of the system or the family of the probability distribution of the time series data. Hence we have not used any prior knowledge of sensors measurement distribution to model EPASAD in our experiment (cf. "Experiments and results" section ).
- *EPASAD is computationally efficient* EPASAD is developed to deploy over real-time CI, which requires processing the streaming measurement. EPASAD is a light-weight framework that produces a decision for measurement in linear time complexity of $O(L)$ in order of lagged vector. EPASAD is tested on a 'Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz' machine with '64-bit Ubuntu 16.04 LTS' operating system and '16 GB' RAM. EPASAD takes 3.6 and 3.0 μsec to generate one result for TE-dataset and C-town datasets, respectively.
- *EPASAD is secure against evasion attack* In Garcia et al. (2017), the authors developed a PLC rootkit that can corrupt the communication route between sensors and SCADA. An attacker can compromise a few communication channels and manipulate them accordingly to hide the structural changes in the normal behavior of any other sensors. In Erba et al. (2020); Biggio and Roli (2018), the authors used this concept to construct an evasion attack against multivariate detectors (Feng et al. 2017; Goh et al. 2017; Taormina and Galelli 2018). In the case of univariate IDS, each sensor is independently modeled. Manipulating a few sensor variables cannot affect any other univariate IDS model. Hence univariate IDS are safer against evasion attacks.
- *EPASAD generates a low false alarm rate* unlike any other nonuniform decision boundary-based model in which low margin sides are volatile to raise a false alarm. EPASAD is motivated to learn a uniform decision boundary, and adding a small slack-value provides a margin of error without compromising accuracy. As a result, EPASAD generated only 3.70% false alarm (cf. Table 2) while testing it for nine months.

Identifying the structural changes in time series data is a classical problem that is useful for detecting irregularities and attacks in a wide range of applications such as an automated vehicle, robotics, UAVs, IoT, etc. Improving the performance of detecting the structural changes in a time series data can also enhance the other applications that will be developed in the future. In addition to using EPASAD in other domains, we would like to extend it as a multivariate model, which can be computationally more suitable for large sensor-connected networks.

**Acknowledgements**
We thank to the C3iHub (Technology Innovation Hub on CyberSecurity and Cyber Security for Cyber-Physical Systems) at IIT Kanpur for partially supportingthis research project.

**Author contributions**
Every author has contributed to the manuscript. All authors readand approved the final manuscript.

**Availability of data and materials**
The data and material of this study are partially available public dataset and partially generated by authors. They are available from the corresponding author upon reasonable request.

## Declarations

**Competing interests**
All the authors declare that they have no competing interests.

Received: 5 December 2022 Accepted: 3 May 2023
Published online: 01 November 2023

**References**

Aoudi W, Almgren M (2020) A scalable specification-agnostic multi-sensor anomaly detection system for IIoT environments. Int J Crit Infrastruct Prot 30(1–8):100377
Aoudi W, Almgren M (2021) A framework for determining robust context-aware attack-detection thresholds for cyber-physical systems. In: 2021 Australasian computer science week multiconference. ACM, Dunedin, New Zealand, pp 1–6
Aoudi W, Iturbe M, Almgren M (2018) Truth will out: departure-based process-level detection of stealthy attacks on control systems. In: ACM SIGSAC conference on computer and communications security. ACM, Toronto, Canada, pp 817–831
Biggio B, Roli F (2018) Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recogn 84:317–331
Broomhead D, King G (1986) Extracting qualitative dynamics from experimental data. Phys D 20(2–3):217–236
Cardenas A, Amin S, Lin Z, Huang Y, Huang C, Sastry S (2011) Attacks against process control systems: risk assessment, detection, and response. In: 6th ACM symposium on information, computer and communications security. ACM, Hong Kong, pp 355–366
CSIS: Significant cyber incidents (2022), https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents, accessed: 03/04/2022
Di Pinto A, Dragoni Y, Carcano A (2018) Triton: the first ICS cyber attack on safety instrument systems. In: Proc. Black Hat USA, vol. 2018. Black Hat, USA, pp 1–26
Dong Q, Yang Z, Chen Y, Li X, Zeng K (2017) Anomaly detection in cognitive radio networks exploiting singular spectrum analysis. In: International conference on mathematical methods, models, and architectures for computer network security. Springer, Springer, Warsaw, Poland, pp 247–259
Downs J, Vogel E (1993) A plant-wide industrial process control problem. Comput Chem Eng 17(3):245–255
Dutta A.K, Mukhoty B, Shukla S.K (2021) Catchall: a robust multivariate intrusion detection system for cyber-physical systems using low rank matrix. In: Proceedings of the 2th Workshop on CPS &IoT security and privacy, pp 47–56
Elsner J, Tsonis A (2013) Singular spectrum analysis: a new tool in time series analysis. Springer Science & Business Media, New York USA
Erba A et al (2020) Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In: Annual computer security applications conference. ACM, Austin, USA, pp 480–495
Falliere N, Murchu L, Chien E (2010) W32.Stuxnet dossier. Tech. rep., White paper, Symantec Corp., Security Response
Feng C, Li T, Chana D (2017) Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks. In: 47th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, Denver, US, pp 261–272
Gao X, Hou J (2016) An improved SVM integrated GS-PCA fault diagnosis approach of Tennessee Eastman process. Neurocomputing 174(Part B):906–911
Garcia L, Brasser F, Cintuglu M, Sadeghi A, Mohammed O, Zonouz S (2017) Hey, my malware knows physics! attacking PLCs with physical model aware rootkit. In: NDSS. NDSS, San Diego, USA, pp 1–15
Goh J, Adepu S, Tan M, Lee Z (2017) Anomaly detection in cyber physical systems using recurrent neural networks. In: 18th international symposium on high assurance systems engineering. IEEE, Singapore, pp 140–145
Golyandina N, Korobeynikov A (2014) Basic singular spectrum analysis and forecasting with R. Comput Stat Data Anal 71:934–954
Golyandina N, Nekrutkin V, Zhigljavsky A (2001) Analysis of time series structure: SSA and related techniques. CRC Press, Boca Raton, Florida
Golyandina N, Zhigljavsky A (2013) Singular spectrum analysis for time series. Springer Science & Business Media, Berlin, Germany
Guan Y, Ghorbani A, Belacel N (2003) Y-Means: a clustering method for intrusion detection. In: Canadian conference on electrical and computer engineering. IEEE, Montreal, Canada, pp 1083–1086
Hadžiosmanović D, Sommer R, Zambon E, Hartel P (2014) Through the eye of the PLC: semantic security monitoring for industrial processes. In: 30th annual computer security applications conference. ACM, New Orleans, USA, pp 126–135
Hansen P, Mladenović N (2001) J-Means: a new local search heuristic for minimum sum of squares clustering. Pattern Recogn 34(2):405–413
Hassani H (2010) A brief introduction to singular spectrum analysis. Tech. rep, Cardiff School of Mathematics
Hu W, Hu W, Maybank S (2008) Adaboost-based algorithm for network intrusion detection. IEEE Trans Syst Man Cybern Part B (Cybern) 38(2):577–583
Jeon H, Eun Y (2019) A stealthy sensor attack for uncertain cyber-physical systems. IEEE Internet Things J 6(4):6345–6352
Kiss I, Genge B, Haller P (2015) A clustering-based approach to detect cyber attacks in process control systems. In: 13th international conference on industrial informatics. IEEE, Cambridge, UK, pp 142–148
Kovacevic A, Nikolic D (2015) Cyber attacks on critical infrastructure: review and challenges. In: Handbook of research on digital crime, cyberspace security, and information assurance, pp 1–18
Lee R, Assante M, Conway T (2016) Analysis of the cyber attack on the Ukrainian power grid. Electr Inf Sharing Anal Center (E-ISAC) Defense Use Case 388:1–29
Lee RM, Assante MJ, Conway T (2014) German steel mill cyber attack. Ind Control Syst 30(62):1–15
Liang G, Zhao J, Luo F, Weller SR, Dong ZY (2016) A review of false data injection attacks against modern power systems. IEEE Trans Smart Grid 8(4):1630–1638
Lichman M et al (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
Mathur A, Tippenhauer N (2016) SWaT: a water treatment testbed for research and training on ICS security. In: International workshop on cyber-physical systems for smart water networks (CySWater). IEEE, Vienna, Austria, pp 31–36
Mo Y, Sinopoli B (2015) On the performance degradation of cyber-physical systems under stealthy integrity attacks. IEEE Trans Autom Control 61(9):2618–2624
Mohammad Y, Nishida T (2011) On comparing SSA-based change point discovery algorithms. In: IEEE/SICE international symposium on system integration (SII). IEEE, Kyoto, Japan, pp 938–945
Moskvina V, Zhigljavsky A (2003) Change-point detection algorithm based on the singular-spectrum analysis. Commun Stat Simul Comput 32:319–352
Nader P, Honeine P, Beauseroy P (2014) $l_p$-norms in one-class classification for intrusion detection in SCADA systems. IEEE Trans Ind Inf 10(4):2308–2317
Shoukry Y, Martin P, Yona Y, Diggavi S, Srivastava M (2015) PyCRA: physical challenge-response authentication for active sensors under spoofing attacks. In: 22nd ACM SIGSAC conference on computer and communications security. ACM, Denver, USA, pp 1004–1015
Smith RS (2015) Covert misappropriation of networked control systems: presenting a feedback structure. IEEE Control Syst Mag 35(1):82–92

Stouffer K, Pillitteri V, Lightman S, Abrams M, Hahn A (2015) Guide to industrial control systems (ICS) security–Rev. 2. Tech. Rep. 82, NIST Special Publication
Taormina R, Galelli S (2018) Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems. J Water Resour Plan Manag 144(10):04018065 (1–15)
Taormina R, Galelli S, Tippenhauer N, Salomons E, Ostfeld A (2017) Characterizing cyber-physical attacks on water distribution systems. J Water Resour Plan Manag 143(5):04017009 (1–12)
Taormina R et al (2018) Battle of the attack detection algorithms: disclosing cyber attacks on water distribution networks. J Water Resour Plann Manag 144(8):04018048 (1–11)
Teixeira A, Shames I, Sandberg H, Johansson KH (2012) Revealing stealthy attacks in control systems. In: 2012 50th Annual Allerton conference on communication, control, and computing (Allerton). IEEE, pp 1806–1813
Terai A, Chiba T, Shintani H, Kojima S, Abe S, Koshijima I (2018) Intrusion detection method for industrial control systems using singular spectrum analysis. WIT Trans Eng Sci 121:197–208
Urbina D, Giraldo J, Cardenas A, Valente J, Faisal M, Tippenhauer N, Ruths J, Candell R, Sandberg H (2016) Survey and new directions for physics-based attack detection in control systems. Tech. rep., National Institute of Standards and Technology
Vautard R, Ghil M (1989) Singular spectrum analysis in nonlinear dynamics, with applications to paleoclimatic time series. Phys D 35(3):395–424
Zheng X, Julien C, Kim M, Khurshid S (2015) Perceptions on the state of the art in verification and validation in cyber-physical systems. IEEE Syst J 11(4):2614–2627
Zhu J, Ge Z, Song Z (2017) Distributed parallel PCA for modeling and monitoring of large-scale plant-wide processes with big data. IEEE Trans Industr Inf 13(4):1877–1885

## Publisher's Note