Recursive Histogram Tracking-Based Rapid Online Anomaly Detection in Cyber-Physical Systems

Ratnesh Kumar, *Fellow, IEEE*, Ramij Raja Hossain, *Graduate Student Member, IEEE*, Soumyabrata Talukder, *Graduate Student Member, IEEE*, Amit Jena, *Student Member, IEEE*, and Alaa T. Al Ghazo, *Member, IEEE*

***Abstract*—Prompt online detection of anomalies induced by malicious attacks enhances the efficacy of real-time operation and mitigation of attack, an indispensable part of any cyber-physical system (CPS) management. This article proposes a novel *online* rapid detection scheme that continuously monitors the data packet stream and infers the sequence of probability distributions, estimated as histograms, and alerts when a change in the histogram is detected, reporting both the attack as well as an estimate of its instant of commencement. A statistical data-driven attack model is proposed and employed that is general enough to represent two ubiquitous types of attacks on CPS: 1) replay and 2) bias-injection. The proposed detection framework relies on the fact that CPSs possess well-defined dynamics that are affected by quasistationary noise, which allows the histogram sequences of the system data packets to converge (to different distributions under the presence of the attack versus the absence of attack). The proposed online scheme detects an attack, and estimates the attack commencement time by relying on the computed distance between real-time estimated histogram versus *apriori* learned nominal histogram. Our formulation further sheds light on two different attack initiation-time-based subcases, "early" (attack starts before sufficient data of nominal behavior was collected to allow its histogram sequence to be closer to its nominal value) versus "late." The designed algorithm of our scheme has linear time complexities in the dimension of data packets and algorithm parameters, which makes it suited for rapid detection. The proposed algorithm is implemented and validated on two real supervisory control and data acquisition system datasets, where a low detection delay demonstrates the effectiveness of the scheme.**

***Index Terms*—Anomaly detection, bias-injection attack, data encoding, online detection, replay attack.**

I. Introduction

CYBER-PHYSICAL systems (CPSs) hold prime importance in several sectors of modern society. Such systems can be described as a congregation of computing, networking, and physical processes. Many essential infrastructures, such as smart grid [1], medical monitoring [2], and water irrigation systems [3], are examples of CPSs. CPS operations rely on the coordination of several sensor and actuator components that continuously interact with the physical system being monitored, and send feedback signals through the network to the computation layer, aiding the overall management and control. The inherent complexities of each of the individual components as well as their intertwined dependencies make the CPSs vulnerable to several security attacks, which can corrupt the system data leading to catastrophic outcomes. Severity and stealthiness of these attacks depend on the resources of attacker/adversary and its knowledge of the target. Among the existing types of attack schemes, manipulating network data to tamper the information supplied is quite prevalent, where a common practice is to inject malicious data packets masquerading as legit ones or replay the existing data packets, and gradually corrupt an underlying process. Another means of security attacks is via "structural" modifications of the measurement data obtained through the sensors deployed at the physical system. Such illegal injected or structurally modified data packets are anomalies, whose timely detection is of paramount importance for resilient operation of CPSs [4] given their dependence on feedback data.

Several existing approaches utilize certain models of a CPS. For example, [5] proposed an attack detection scheme for a replay attack launched under a steady operating condition, by relying on linearlized dynamics for typical state estimation. Liu *et al.* [6] ultimately formulated a matrix separation problem for attack detection. Ye *et al.* [7] analyzed the residual-based detection scheme for CPS with linear dynamics. The works in [8] and [9] utilize deviation from modeled dynamics for anomaly detection, whereas to distinguish an exogenous attack from a component failure, a classifier-based method was added on top of the base anomaly detector. All these methods require explicit knowledge of the system model, which is not always practical. Moreover, in many cases, the dynamics of the system is not well approximated by either a linearized dynamic model or an automaton, which makes the attack identification error prone. Furthermore, such

Manuscript received 19 July 2021; accepted 2 February 2022. Date of publication 17 February 2022; date of current version 17 October 2022. This work was supported in part by U.S. National Science Foundation under Grant NSF-CCF-1331390, Grant NSF-ECCS-1509420, Grant NSF-PFI-1602089, and Grant NSF-CSSI-2004766. This article was recommended by Associate Editor K. G. Vamvoudakis. *(Corresponding author: Ratnesh Kumar.)*

Ratnesh Kumar, Ramij Raja Hossain, Soumyabrata Talukder, and Amit Jena are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 USA (e-mail: rkumar@iastate.edu; rhossain@iastate.edu; talukder@iastate.edu; amitjenamath@gmail.com).

Alaa T. Al Ghazo is with the Department of Mechatronics Engineering, The Hashemite University, Zarqa 13133, Jordan (e-mail: alghazo@hu.edu.jo).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3150304.

Digital Object Identifier 10.1109/TSMC.2022.3150304

methods require a new model each time the system setting changes.

To circumvent the limitations of model-based approaches, few statistics and machine learning-based data-driven approaches have also been explored [10]–[14], utilizing both supervised and unsupervised-based learning methods. But the success of these methods is largely dependent on the generation of suitable training data combining both normal data and data under malicious attacks, and this will affect the attack detection if the real-time attack comes from a different data distribution compared to the one used for training.

Among the existing works on histogram-based anomaly detection technique, [15] constructs histograms of different header features of the data-packets offline, and those are clustered after mapping onto a metric space, and finally, an anomaly is detected by comparing the distance of the histogram of the online observed data to the learned clusters. The approach in [16] monitors the payload features of the data packets offline for constructing a histogram of each feature. A scalar score is assigned to a data observed online based on the probability of its payload features from the respective normalized histograms, and the data are declared anomalous if the score exceeds an empirical threshold. It is important to note that in both [15] and [16], the features are assumed uncorrelated.

In this article, we propose a data-driven approach and a novel algorithm to detect any anomaly that may be present in the system data stream in an online manner by estimating and monitoring the histogram of probability distribution of data packets over time. Online anomaly detection is necessary in real-time attack detection and mitigation in CPSs, which rely on uncorrupted feedback data for safe and stable operation. Our online approach thereby is novel compared to the prior offline attack detection, such as in case of an electricity theft, the distribution of meter measurement readings over a certain period of time differs from that without any theft [17].

The contributions of our work are as follows.

1) A completely *online* approach is proposed, where using streaming time-series data, we learn online histogram of the probability distribution of data, and detect an attack that causes a change in the probability distribution of data packets (under persistence of attack).
2) Robust notions of "trend reversal" and "convergence" for a datastream are introduced and utilized.
3) We designed an algorithm of linear time complexity in the dimension of data, number of histogram bins used, and window length used in deciding "trend-reversal" or "convergence" for attack detection, making the scheme rapid, allowing for immediate mitigation response.
4) We estimate the *time of attack* commencement that helps to unveil the period of corrupted data.
5) The proposed algorithm is implemented on datasets obtained from a real-world supervisory control and data acquisition (SCADA) system of iTrust Lab [18] to validate its effectiveness and noise robustness, and also to demonstrate its low detection delay.

II. Online Anomaly Detection Problem

In this section, we first represent the probability distribution of data packets in the form of normalized (unit mass) histograms. Next, we describe the procedure to compute such histograms of nominal or under attack data. Subsequently, we describe the adversary data for various attack classes and formulate their respective histograms. Next, a generalized attack model (GAM) is suggested to which all of the considered attack cases conform. The section ends with a mathematical formulation of the online anomaly detection problem, whose solution is then discussed in the subsequent sections.

A. Histogram Computation from System Data

For simplicity of explanation, we begin with the case of scalar data, and later in Section V, we explain the extension to the multidimensional case. The system data packet at time $i \geq 1$ is denoted by $y_i \in \mathbb{R}$. By a time instant $i \geq 1$, an observer would have witnessed the history of data: $Y_i := \{y_j \mid j \in \{1, 2, \ldots, i\}\}$, using which it can construct a normalized histogram, denoted H_i, over a fixed-set of histogram bins $([b_0, b_1), \ldots, [b_{m-1}, b_m))$, with $\bigcup_{k=1}^{m} [b_{k-1}, b_k) \subset \mathbb{R}$. Here, H_i represents the sample distribution histogram of the governing stationary process at time instant i and is defined as follows:

$$H_i := \left(p_1^i, p_2^i, \ldots, p_m^i\right) \tag{1}$$

with

$$p_k^i := \frac{\sum_{j=1}^{i} I\left(y_j \in \left[b_{k-1}, b_k\right)\right)}{i} \tag{2}$$

where $I(.)$ is the standard binary indicator function, and p_k^i computes the fraction of data from this history Y_i that falls in the kth bin. It can be noted that

$$\forall j : 0 \leq p_k^i \leq 1, \text{ and } \sum_{k=1}^{m} p_k^i = 1.$$

Using the streaming time series of data, one can recursively generate a sequence of histograms, $\{H_i\}_{i \in \mathbb{N}}$ through an iterative process described as

$$H_1 := \delta_{y_1}; \quad H_i := \frac{1}{i}\left((i-1)H_{i-1} + \delta_{y_i}\right), i \geq 1 \tag{3}$$

where δ_{y_i} represents a degenerate histogram having the entire probability mass concentrated at the bin containing y_i, i.e.,

$$\delta_{y_i} := \left(p_1^i, \ldots, p_m^i\right), \text{ with } p_k^i := \begin{cases} 1, & \text{if } y_i \in \left[b_{k-1}, b_k\right) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

A visual depiction of the iterative histogram computation process (3) is depicted in Fig. 1.

B. Modeling of Attack Data

We consider two frequently encountered "man-in-middle" attacks on CPSs, namely, replay attack and bias injection attack. Note the latter is a case of structural modification of data packets, namely, introduction of a bias. Denoting the attack data at time instant $i \geq 1$ as y_i^a, we define its value for the three attack scenarios, where the indexing is chosen

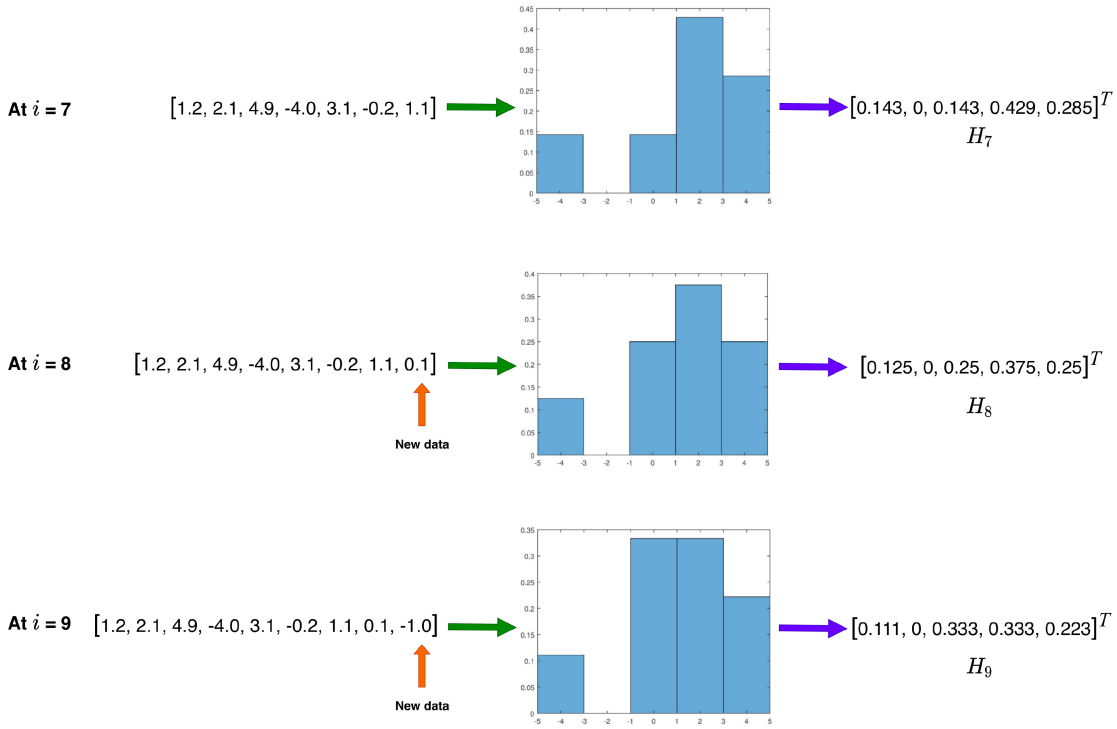

Fig. 1. Sample visualization of the histogram computation process. The 5 fixed bins are $([-5,-3),[-3,-1),[-1,1),[1,3),[3,5))$. At each i, an incoming data y_i is received and employed to compute a normalized histogram H_i (drawn in vector form as the output) by updating the earlier historgram H_{i-1} (shown in the middle pictorially); the above diagrams show the case for $i = 7, 8, 9$.

to allow the flexibility of attack commencement at any time point. Then, the attack data stream $y_i^a, i \geq 1$ stimulates an attack histogram sequence $H_i^a, i \geq 1$ as in the case of nominal streaming data.

1) Replay Attack: In general, replay attack is broad in scope, and can be undetectable when meticulously designed [19]. Here, we restrict our scope to the following scenario: the adversary records successively received system data packets from a start time s to an end time e. The attack data then are obtained by overwriting all the system data by the precollected data packets that are repetitively replayed maintaining their recorded sequential order [7]. The corresponding adversarial data is then given by

$$y_i^a := y_{s+i \bmod (e-s+1)}, \text{ if } i \geq 1 \tag{5}$$

where "mod" is the standard modulo operation.

2) Bias Injection Attack: This attack deals with adding a constant bias to the sensor measurement data while being stealthy at the same time [20]–[22]. This can drive the CPS to an unsafe operating region and trigger cascading events, leading to a system blackout. We represent the adversarial data of this case by

$$y_i^a := y_i + y_c, \text{ if } i \geq 1 \tag{6}$$

where y_c is a constant bias that is chosen from the range space of a certain the Jacobian matrix of the CPS. For the derivation of y_c that can spoof "residue detection" and additional insights, [22] can be referred.

C. Generalized Attack Model

We consider an attack that commences at an unknown time instant $i = n$, which needs to be estimated. The commencement of the attack alters the data stream either by switching them with malicious data packets or structurally modifying them. This leads to the change in ensuing histogram sequence of data packets. In such a GAM, the histogram sequence is generated from nominal data stream (for $i < n$) and changes to an attack data stream (for $i \geq n$). We denote the corresponding histogram sequence as $\{H_i^a\}_{i\in\mathbb{N}}$ and note the following equivalence: $H_i^a = H_i$ if $i < n$,. The expression of H_i^a that is valid for all $i \geq 1$ can be given as follows:

$$\begin{aligned} H_i^a &= \frac{\sum_{j=1}^{i} \delta_{y_j^a}}{i} = \frac{\sum_{j=1}^{n-1} \delta_{y_j}}{i} + \frac{\sum_{j=n}^{i} \delta_{y_j^a}}{i} \\ &= \left(\frac{n-1}{i}\right)\left(\frac{\sum_{j=1}^{n-1} \delta_{y_j}}{n-1}\right) + \frac{\sum_{j=n}^{i} \delta_{y_j^a}}{i} \\ &= \left(\frac{n-1}{i}\right) H_{n-1} + \frac{\sum_{j=n}^{i} \delta_{y_j^a}}{i}. \end{aligned} \tag{7}$$

D. Derivation of GAM for the Proposed Attack Cases

Here, we derive the specific forms of $H_i^a, \ i \geq 1$ for each of the attack cases introduced above.

1) Replay Attack: By definition, the attacker captures a fixed number of successively received data packets, $\{y_s, y_{s+1}, \ldots, y_e\}$ and replays them during the attack. The second term on the right-hand side of (7) can then be simplified by utilizing the fact that $\{y_s, y_{s+1}, \ldots, y_e\}$ gets repeated after the attack happens. This allows us to express $H_i^a, i \geq 1$ in a

consolidated form as follows:

$$H_i^a = \begin{cases} \left(\frac{n-1}{i}\right)H_{n-1} + \left(\frac{q_i}{i}\right)\sum_{j=s}^{e}\delta_{y_j} \\ \quad + \left(\frac{1}{i}\right)\sum_{j=s}^{s+r_i-1}\delta_{y_j}, & \text{if } r_i \neq 0 \\ \left(\frac{n-1}{i}\right)H_{n-1} + \left(\frac{q_i}{i}\right)\sum_{j=s}^{e}\delta_{y_j}, & \text{otherwise} \end{cases} \quad (8)$$

where

$$r_i = (i-n+1) \bmod (e-s+1)$$
$$q_i = \frac{i-n+1-r_i}{e-s+1}. \quad (9)$$

2) Bias Injection Attack: In this case, $y_j^a = y_j + y_c$, and hence, (7) can be rewritten to obtain $H_i^a, i \geq 1$ as follows:

$$H_i^a = \left(\frac{n-1}{i}\right)H_{n-1} + \frac{\sum_{j=n}^{i}\delta_{y_j+y_c}}{i}. \quad (10)$$

From the construction of δ_{y_i} and $\delta_{y_i+y_c}$, the following facts are evident.

1) If a constant bias y_c is added to y_i, the corresponding bin of y_i shifts by some units.
2) The units by which the bin is shifted in general depends on y_c, y_i, and δ_{y_i}.

So we can introduce the following equality:

$$\delta_{y_i+y_c} = C_{\delta_{y_i},y_c}\delta_{y_i} \quad (11)$$

where $C_{\delta_{y_i},y_c}$ is an $m \times m$ binary matrix mapping to δ_{y_i} and $\delta_{y_i+y_c}$. As per the definition, y_c remains constant throughout the attack. Hence, from now on we drop y_c from the subscript of the matrix $C_{\delta_{y_i},y_c}$. Accordingly, by employing (11), (10) can be rewritten as

$$H_i^a = \left(\frac{n-1}{i}\right)H_{n-1} + \frac{\sum_{j=n}^{i}C_{\delta_{y_j}}\delta_{y_j}}{i}. \quad (12)$$

E. Online Anomaly Detection Problem Statement

Motivated by the goal of online anomaly detection, the concrete problem statements are formalized as follows.

1) *Online Attack Detection:* Track H_i^a in an online manner and detect if the CPS is inflicted by an attack.
2) *Online Estimation of Attack Point:* If an attack is detected, then estimate the time of attack, i.e., an estimate $\hat{n}$ of n.

III. Stationarity of Distributions and Early Versus Late Attacks

We consider the class of CPSs with quasistationary white noise, which imply that the underlying process is an ergodic Markov process. As noted above, the time series of observations from a CPS is discretized into m disjoint intervals or "bins," $([b_0, b_1), \ldots, [b_{m-1}, b_m))$, where $b_{k-1} \in \mathbb{R}$ and $\bigcup_{k=1}^{m}[b_{k-1}\ b_k) = \mathbb{R}$. Each such bin can be viewed as a "state" of an associated finite state Markov chain. An ergodic m-state Markov chain possesses a unique m-dimensional stationary distribution π [23]. It follows that regardless of the initialization of the Markov chain, the empirical histogram distribution H_i [as defined in (1) and (2)] will converge to π as $i \to \infty$ since the kth element of π represents the fraction of time the chain spends at the kth state [23].

Similar to the previous section, our formulation here is presented for the scalar observed variable (i.e., $q = 1$), while the generalization is covered later in Section V. We show that if the attacks considered are persistent enough, then H_i^a is also convergent, i.e., the following holds:

$$\lim_{i\to\infty} H_i = H; \quad \lim_{i\to\infty} H_i^a = H^a. \quad (13)$$

Next, we proceed to validate that the second equality holds for each of our attack cases.

Theorem 1: It holds that both of the attack cases (replay and bias injection) satisfy (13), i.e., the histogram sequence under attack converges.

Proof:

1) *Replay Attack:* Before delving into evaluating the limiting value of H_i^a, we state the following fact:

$$0 < r_i < e-s+1 \implies \sum_{j=s}^{s+r_i-1}\delta_{y_j} \text{ is a vector having finite norm}$$
$$\implies \lim_{i\to\infty}\left(\frac{1}{i}\right)\sum_{j=s}^{s+r_i-1}\delta_{y_j} = 0.$$

From the above observation, it is easy to note that the limiting value of H_i^a is the same for the case of $r_i = 0$ as well as the case $r_i \neq 0$. So, using the $r_i \neq 0$ case in (8), we compute the limit of H_i^a as follows:

$$\lim_{i\to\infty} H_i^a \overset{(8)}{=} \lim_{i\to\infty}\left(\frac{n-1}{i}\right)H_{n-1} + \lim_{i\to\infty}\left(\frac{q_i}{i}\right)\sum_{j=s}^{e}\delta_{y_j}$$
$$= 0 + \lim_{i\to\infty}\left(\frac{q_i}{i}\right)\sum_{j=s}^{e}\delta_{y_j} \quad (\text{as } H_{n-1} \text{ is a vector of finite norm})$$
$$\overset{(9)}{=} \lim_{i\to\infty}\frac{(i-n+1-r_i)}{(e-s+1)i}\sum_{j=s}^{e}\delta_{y_j}$$
$$= \frac{1}{e-s+1}\sum_{j=s}^{e}\delta_{y_j}. \quad (14)$$

The existence of the limiting value (14) testifies that the replay attack satisfies the property of convergence in (13).

2) *Bias Injection Attack:* To start, we develop a few results that will facilitate the computation of the limiting value of H_i^a in this case. First we express the property (13) as follows:

$$H := \lim_{i\to\infty} H_i = \lim_{i\to\infty}\frac{\sum_{j=1}^{i}\delta_{y_j}}{i}$$
$$= \lim_{i\to\infty}\frac{\sum_{j=1}^{n-1}\delta_{y_j}}{i} + \lim_{i\to\infty}\frac{\sum_{j=n}^{i}\delta_{y_j}}{i}$$
$$= \lim_{i\to\infty}\left(\frac{n-1}{i}\right)H_{n-1} + \lim_{i\to\infty}\frac{\sum_{j=n}^{i}\delta_{y_j}}{i}$$
$$= 0 + \lim_{i\to\infty}\frac{\sum_{j=n}^{i}\delta_{y_j}}{i}. \quad (15)$$

Now, if we partition $\lim_{i\to\infty}[(\sum_{j=n}^{i}\delta_{y_j})/i]$ into summations over individual bins, then we get

$$H = \lim_{i\to\infty}\frac{\sum_{j=n}^{i}\delta_{y_j}}{i} = \sum_{k=1}^{m-1}\left(\lim_{i\to\infty}\frac{1}{i}\sum_{\substack{n\le j\le i\\ y_j\in[b_{k-1},b_k)}}\delta_{y_j}\right). \tag{16}$$

It then follows that:

$$H = (p_1, p_2, \ldots, p_m), \text{ with}$$
$$\forall k \in [1,m] : p_k = \left(\lim_{i\to\infty}\frac{1}{i}\sum_{\substack{n\le j\le i\\ y_j\in[b_{k-1},b_k)}}\delta_{y_j}\right).\mathbb{1}_m \tag{17}$$

where $\mathbb{1}_m$ denotes the column vector of length m with all entries 1, and is multiplied to the row-vector inside the bracketed term in (17) to be able to add all its entries (which are all zeros except at the kth location). Next, we proceed to compute the limiting value of H_i^a for this attack case as follows:

$$\lim_{i\to\infty}H_i^a = \left(\frac{n-1}{i}\right)H_{n-1} + \frac{\sum_{j=n}^{i}C_{\delta_{y_j}}\delta_{y_j}}{i} = 0 + \lim_{i\to\infty}\frac{\sum_{j=n}^{i}C_{\delta_{y_j}}\delta_{y_j}}{i} = \lim_{i\to\infty}\frac{\sum_{j=n}^{i}C_{\delta_{y_j}}\delta_{y_j}}{i}. \tag{18}$$

Now, similar to (17), we have

$$H^a = \left(p_1^a, p_2^a, \ldots, p_m^a\right), \text{ with}$$
$$\forall k \in [1,m] : p_k^a = \left(\lim_{i\to\infty}\frac{1}{i}\sum_{\substack{n\le j\le i\\ y_j\in[b_{k-1},b_k)}}C_{\delta_{y_j}}\delta_{y_j}\right).\mathbb{1}_m. \tag{19}$$

This establishes that even under the bias injection attack, the histogram sequence converges and the property (13) holds. ■

A. Early Versus Late Attack Cases

In order to compare the observed sequence of histogram H_i^a with the histogram of nominal (nonattack) case H, we consider the notion of histogram norm, which we denote as $\|.\| : \mathbb{R}^m \to \mathbb{R}^+ \cup \{0\}$ (e.g., ℓ_1, ℓ_2, and ℓ_∞ norm [24]). It is evident that when there is no attack (i.e., attack time point $n = \infty$), $H_i^a = H_i$ for all $i \ge 1$, and so under (13), H_i^a converges to H. On the other hand, when there is an attack, then by (13) again, H_i^a converges to H^a, i.e.,

$$\lim_{i\to\infty}\left\|H_i^a - H\right\| = \begin{cases} 0, & \text{if there is no attack} \\ \|H^a - H\| \neq 0, & \text{if an attack starts at } i = n. \end{cases} \tag{20}$$

Following the above observation, we treat $\|H_i^a - H\|$ as an indicator of the presence or absence of an attack. Furthermore, guided by (20), we define two different cases of attack, based on its initiation time.

Note if there is no attack, then from (20), $\|H_i^a - H\|$ converges toward zero, and in doing so, it falls below the $\|H_a - H\| \neq 0$ value at some point. If the attack occurs after this point, then it is termed a "late attack" and otherwise, it is termed an "early attack." [See Fig. 4(a) and (c) for the case of early attack, and Fig. 4(b) and (d) for the case of late attack, which plot $d_i := \|H_i^a - H\|$ against i.]

Accordingly, we define the two attack cases as follows.

1) *Early Attack:*

$$n < \min\left\{i \in \mathbb{N} \;\middle|\; d_i := \left\|H_i^a - H\right\| < \left\|H^a - H\right\|\right\}. \tag{21}$$

2) *Late Attack:*

$$n \ge \min\left\{i \in \mathbb{N} \;\middle|\; d_i := \left\|H_i^a - H\right\| < \left\|H^a - H\right\|\right\}. \tag{22}$$

It is evident from the case of late attack (22), where $d_i = \|H_i^a - H\|$ has fallen below $\|H^a - H\|$ prior to the attack, a "trend reversal" will be manifested in $d_i = \|H_i^a - H\|$ (goes from decreasing toward zero to increasing toward $\|H^a - H\|$) in order to converge from a below $\|H^a - H\|$ value to the $\|H^a - H\|$ value. In contrast, no such trend reversal in $\|H_i^a - H\|$ can be observed in the early case (21). This separates the two attack cases.

We propose our solution to the online attack detection as well as the attack point estimation problems taking the above factors into account, as described in the next section.

IV. Attack Detection and Attack Point Estimation

We impose the following definitions, which are required for the solution process to be described:

$$d_i := \left\|H_i^a - H\right\|, \quad d_i^{\text{mov}} := \frac{\sum_{j=i-l+1}^{i} d_j}{l}. \tag{23}$$

Here, d_i is simply a shorthand for $\|H_i^a - H\|$ that we track, and d_i^{mov} is the moving average of d_i over the latest window of length l (a parameter of the proposed detection algorithm). Since the measurement data are noisy, the convergence of d_i is not monotonic, and a "robust" notion of convergence is needed, which we introduce as follows.

Definition 1: The variable d_i is said to have "converged" when its value differs from all its past w values by no more than a small tolerance bound ϵ_{conv}. The "convergence time" n_{conv} is defined to be the earliest time instant when this occurs

$$n_{\text{conv}} := \min\left\{i \in \mathbb{N} \middle| i > w \;\forall j \in \{1,\ldots,w\} \; \left|d_i - d_{i-j}\right| < \epsilon_{\text{conv}}\right\}. \tag{24}$$

Here, w is another time window (different from l and yet another algorithmic parameter) that is used in deciding the convergence.

An interesting fact follows for the setting of our problem that there exists of a time point of trend reversal (where d_i becomes increasing from decreasing) in the case of "late attack." Again owing to noisy setting, a trend reversal must be defined robustly as we do next.

Definition 2: A "trend-reversal" in d_i is said to have occurred at a certain instant if d_i switches from being smaller than its moving average to being larger than its moving average, i.e.,

$$\exists i \geq n : \left(d_i < d_{i-1}^{\text{mov}} - \epsilon_{tr}\right) \wedge \left(d_{i+1} > d_i^{\text{mov}} + \epsilon_{tr}\right) \quad (25)$$

where ϵ_{tr} represents a given tolerance constant to check for trend reversal.

We first prove the existence of a finite $i \geq n$ where a trend reversal occurs for the case of "late attack."

Theorem 2: In the case of a "late attack" (22), a point of trend reversal follows the attack point, i.e., (25) holds.

Proof: Suppose for contradiction

$$\left[\forall i \geq n : d_{i+1} \leq d_i^{\text{mov}} + \epsilon_{tr}\right] \Leftrightarrow \left[\forall i \geq n : d_i^{\text{mov}} > d_{i+1}\right]. \quad (26)$$

Then, summing over l consecutive indices, we have for all $i \geq n$

$$\sum_{j=0}^{l-1} d_{i+j}^{\text{mov}} > \sum_{j=0}^{l-1} d_{i+1+j} = l d_{i+l}^{\text{mov}}$$

where the equality follows from the definition of moving average (23). This then implies

$$\frac{1}{l}\sum_{j=0}^{l-1} d_{i+j}^{\text{mov}} > d_{i+l}^{\text{mov}}. \quad (27)$$

From (27), it follows that for any $i \geq n$, the moving average at $i+l$ is smaller than the average of all the past l moving averages, meaning that the moving average decreases on average. Furthermore, since the moving averages are lower bounded by zero (since d_i's are lower bounded by zero), it follows that the moving average d_i^{mov} will eventually converge to zero. But this contradicts (13) [also see (20)] that $\lim_{i\to\infty} H_i^a = H^a$, which implies that

$$\lim_{i\to\infty} d_i^{\text{mov}} = \lim_{i\to\infty} d_i = \lim_{i\to\infty} \left\|H_i^a - H\right\| = \left\|H^a - H\right\| \neq 0. \quad \blacksquare$$

Given the existence of trend reversal in the case of late attack, next we introduce the variable n_{tr} to define the earliest instant where the trend reversal occurs.

Definition 3: Taking into account the theorem above, we define the "time point of trend reversal" as the following:

$$n_{tr} := \min\left\{i \in \mathbb{N} \middle| d_i < d_{i-1}^{\text{mov}} - \epsilon_{tr}, d_{i+1} > d_i^{\text{mov}} + \epsilon_{tr}\right\}.$$

Now, we present our solution to the two problems related to online anomaly detection that we formulated above.

1) *Online Attack Detection:* Track $d_i = \|H_i^a - H\|$ to the time point of convergence n_{conv} and check if $d_{n_{\text{conv}}} > \epsilon$, where ϵ is another tolerance constant in the range $(0, \|H^a - H\|)$. If the condition is true (meaning d_i converges to a value away from zero), then confirm the detection of an attack, and otherwise, declare that there is no attack. Report the value of the attack detection indicator variable ζ, defined as follows:

$$\zeta := \begin{cases} 1, & \text{if } d_{n_{\text{conv}}} > \epsilon \\ 0, & \text{otherwise.} \end{cases} \quad (28)$$

Algorithm 1 Data Encoding via PH

```
function PH(y_i)
    h ← 0, i ← 1
    S ← TS(y_i)                          ▷ Text string conversion
    q̂ ← length(S)   ▷ computation of total characters of S
    for j ≤ q̂ do
        h ← T[ h XOR U(S[j]) ]
              ▷ U(S[j]) is the Unicode of j^th character of S
    end for
    return h
end function
```

2) *Online Estimation of Attack Point:* Track d_i to the time point of convergence n_{conv}, noting whether a trend reversal time point n_{tr} occurred first. Note down n_{conv} and if applicable n_{tr} values, and report the estimate of attack point $\widehat{n}$ as

$$\widehat{n} := \begin{cases} n_{tr}, & \text{if } \zeta = 1 \text{ and a trend reversal detected} \\ n_{\text{conv}}, & \text{if } \zeta = 1 \text{ and no trend reversal detected} \\ \infty, & \text{if } \zeta = 0. \end{cases} \quad (29)$$

Here, the first two cases are when there is an attack, with the first case for late attack versus the second case for early attack. The third case is for no attack.

V. Algorithm Presentation and Implementation

In this section, we present our approach to the more general case of multidimensional input data, i.e., $y_i \in \mathbb{R}^q$ by first implementing a dimensionality reduction. The reason for introducing such compression is to attain efficiency of online implementation by making it independent of the dimension of the data. Traditional dimensionality reduction techniques, such as PCA, Isomap, and t-SNE, require the entire dataset to be available, which is not possible for online computations. So, we use a simple and fast data encoding technique, namely, Pearson hashing (PH), as explained next.

A. Pearson Hashing and Data Encoding

PH in our case is an 8-bit hash function to encode data-packet strings to an integer in the range [0, 255]. It is simple to implement, computationally fast, and quite collision resistant (the event of different inputs encoded into identical outputs is less likely) [25]. A simple look up table denoted as T containing a randomized permutation of all the integers in [0, 255] is employed in the encoding process. During our encoding process, we first convert y_i to a text string containing $\hat{q} \geq q$ characters by conjoining all of the entries of y_i (including the negative and decimal signs). This is followed by the mapping of the text string to an integer in [0, 255] with a time complexity of $\mathcal{O}(q)$, where recall that q is dimensionality of each data in the observed time series. The data sample encoding algorithm PH is presented in Algorithm 1 in pseudocode form, whereas a visualization of the encoding process is presented in Fig. 2.

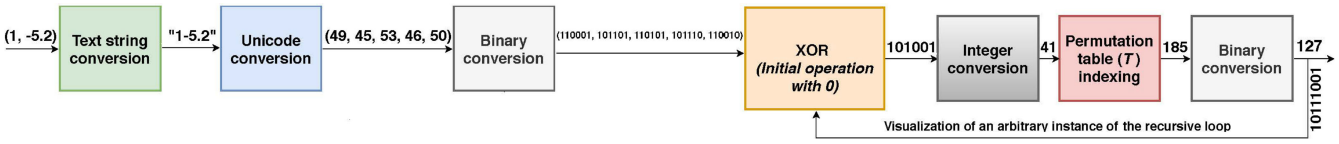

Fig. 2. Block diagram of a sample encoding process of multidimensional data using PH.

Algorithm 2 HIST

function HIST(y_i, H_{i-1})
  $(p_1^{i-1}, p_2^{i-1}, \ldots, p_m^{i-1}) \leftarrow (i-1)H_{i-1}$
  $[b_{g-1}, b_g) \leftarrow \text{BS}\big(\text{PH}(y_i), \big([b_0, b_1), \ldots, [b_{m-1}, b_m)\big)\big)$
    ▷ Binary search operation
  $p_g^i \leftarrow p_g^{i-1} + 1$ ▷ Bin mass update
  $H_i \leftarrow \dfrac{(p_1^i, p_2^i, \ldots, p_m^i)}{i}$
    ▷ Normalized histogram computation
  return H_i
end function

Algorithm 3 Stationary HIST (offline)

Input: Real time data without attack (y_i), threshold (ϵ_{st})
Output: Stationary histogram (H)
$i \leftarrow 1$
$H_0 \leftarrow (0, 0, \ldots, 0)$
do
  $H_i \leftarrow \text{HIST}(y_i, H_{i-1})$ ▷ Function call to Algorithm 2
  $i \leftarrow i + 1$
while $\|H_i - H_{i-1}\| > \epsilon_{st}$ ▷ Convergence criterion
$H \leftarrow H_i$
End

B. Histogram Computation Algorithm

Computation of H_i is performed in a streaming manner where upon receiving the encoded data $\text{PH}(y_i)$, an appropriate bin that it belongs to is identified using the standard binary search (BS) algorithm. Subsequently, the corresponding bin mass is updated and the histogram is normalized as in (3). We present the corresponding algorithm histogram computation algorithm (HIST) in a pseudocode form in Algorithm 2, whose time complexity is $\mathcal{O}(q+m)$ where q and m are the data dimension and the number of bins, respectively.

C. SZscore Algorithm for Minima Detection

The trend reversal forms a minima in the trend of d_i, which we detect in an online manner by implementing the SZscore algorithm [26], an online optima detection algorithm for noisy data. The SZscore algorithm continuously monitors d_i and notifies if a trend reversal is found (by checking whether d_i switches from being below to being above its moving average d_i^{mov}), i.e., for the trend reversal detection, SZscore relies on the definition of n_{tr} and has a time complexity of $\mathcal{O}(l)$ ([26]), where recall that l is the length of the window used in the moving average.

D. Overall Attack Detection Algorithm

The attack detection algorithm works online and tracks the value of d_i to the time point of convergence to detect any attack occurrence. The computation of d_i relies on the online estimated histogram H_i^a (computed using HIST) and the stationary distribution H of nominal nonattack data, which is computed offline, prior to the online attack detection, using Algorithm 3. The attack detection algorithm further uses the SZscore algorithm to detect any trend reversal of d_i. We present the overall algorithm MAIN in pseudocode form in Algorithm 4. The time complexity of the algorithm at each instant is $\mathcal{O}(q + m + \max(l, w))$, which testifies that the algorithm is computationally fast in nature.

Algorithm 4 Attack Detection Algorithm (MAIN)

Input: Real time data (y_i), Stationary histogram (H)
Output: Attack detection variable (ζ), Estimated attack initiation time ($\hat{n}$)
Parameter: n_{tr}, n_{conv}
$i \leftarrow 1$, $n_{tr} \leftarrow 0$
$H_0^a \leftarrow (0, 0, \ldots, 0)$
$d_0 \leftarrow \|H_0^a - H\|$
do
  $H_i^a \leftarrow \text{HIST}(y_i, i-1, H_{i-1}^a)$ ▷ Histogram comp.
  $d_i \leftarrow \|H_i^a - H\|$ ▷ Norm computation
  $n_{tr} \leftarrow \text{SZscore}(d_i, i)$ ▷ d_i Monitoring
  $i \leftarrow i + 1$
while $\sum_{j=1}^{w} |d_i - d_{i-j}| > w\epsilon_{conv}$ ▷ Convergence criterion
$n_{conv} \leftarrow i$
if $d_i > \epsilon$ **then**
  $\zeta \leftarrow 1$ ▷ Attack detection confirmation
  if $n_{tr} \neq 0$ **then**
    $\hat{n} \leftarrow n_{tr}$ ▷ Trend reversal found
  else
    $\hat{n} \leftarrow n_{conv}$ ▷ No trend reversal
  end if
else
  $\zeta \leftarrow 0$ ▷ No attack
end if
End

E. Digit/Component-Based Algorithm for Comparison

While the PH is one way to achieve dimensionality reduction, another way is to focus on the distribution of an individual component of the vector data. This way of tracking and evaluating data finds its root in the old but classic "*Benford's law*" [27] that suggests that in many applications involving numerical data, the probability of the leading digit

being a number $k \in \{1, \ldots, 9\}$ follows the distribution:

$$Pr(k) = \log_{10}(k+1) - \log_{10}(k) = \log_{10}\left(1 + \frac{1}{k}\right). \quad (30)$$

There exist several data analytics techniques based on this observation, and are also quite effective in certain applications, such as fraud detection and stock market analysis [28].

We take inspiration from this paradigm and investigate whether an attack induces any alteration in the probability distribution sequence of any of the individual components of the observed multidimensional data. The workflow based on this method follows Algorithm 4 but instead of using encoding for dimensionality reduction, it simply considers one of the components of the vector data. In parallel, we compute the histograms of all the individual components in the incoming q-dimensional data packets. Following our definition in (29), we compute the estimates of attack instants ($\hat{n}$) for all the individual components. The earliest attack instant among that of all the components is then reported as the estimated time of attack, and is used for comparison with that of our attack detection algorithm involving dimensionality reduction, i.e., Algorithm 4.

VI. Numerical Experiments and Discussion

A. Description of Datasets and Algorithmic Parameters

We collected two real SCADA system datasets, obtained from the iTrust Lab [18] and synthetically incorporated attack data using "SIMATIC S7-PLCSIM" [29]. The iTrust Lab has a two-level network with the control room and an engineering laptop at the first level, and the PLCs at the second level. The network has a single central router that connects to all the devices, which is also the point where the data are collected. Note that the datasets we use in this work were also analyzed in our work on automatic attack graph generation [30]. These two datasets were named as dataset1 and dataset2, respectively. Each of the datasets comprises of 800 number of 20 dimensional data packets. Furthermore, in each of the three attack cases, we include two subcases pertaining to early and late attacks as described in (21) and (22), respectively. For early versus late attacks, we set the attack start time n as 1 versus 100, respectively. In the replay attack, the replayed sequence during attack is the one recorded from $i = 41$ to $i = 50$ under no attack setting. For the bias injection attack, we sampled a 20 dimensional bias vector from a uniform distribution on the support $[0, 100]^{20}$. In specific, the bias vector that got sampled is

$$y_c = [71, 21, 5, 7, 41, 6, 13, 4, 7, 5, 5, 78, 2, 26, 99, 7, 96, 98, 45, 3].$$

The length of time windows for computation of d_i^{mov} and n_{conv} is set as: $l = 5$ and $w = 10$. The tolerance threshold values used in the algorithm are: $\epsilon_{st} = 0.01$, $\epsilon_{tr} = 0.01$, $\epsilon_{\text{conv}} = 0.001$, and $\epsilon = 0.05$.

B. Algorithmic Performance Results

We first demonstrate the convergence of $\|H_i - H_{i-1}\|$ (computed by Algorithm 3) for each dataset as shown in Fig. 3. These plots empirically conform to the fact that in the encoded space, each dataset is convergent [and thereby satisfies (13)]. Then, plots of d_i with respect to time points (i) for each of the attack cases and corresponding subcases are presented in Figs. 4 and 5, respectively, where a trend reversal is evident for the late attack cases, whereas a convergence is seen in all cases, as to be expected per our analysis.

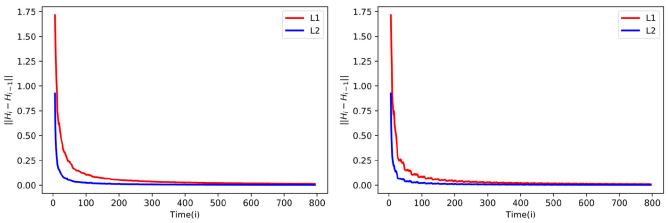

Fig. 3. Convergence of $\{H_i\}_{i\in\mathbb{N}}$ to the stationary distribution H in the encoded space under ℓ_1 (red) versus ℓ_2 (blue) norms. (a) Dataset1. (b) Dataset2.

TABLE I
Performances of Detection for Replay Attack (Encoding Based)

Attack Dataset	n_{tr} (ℓ_1)	n_{conv} (ℓ_1)	$\hat{n}-n$ (ℓ_1)	n_{tr} (ℓ_2)	n_{conv} (ℓ_2)	$\hat{n}-n$ (ℓ_2)
1: EA	–	12	11	–	25	24
1: LA	104	400	4	102	203	2
2: EA	–	12	11	–	34	33
2: LA	102	331	2	155	102	2

TABLE II
Performances of Detection for Bias-Injection Attack (Encoding Based)

Attack Dataset	n_{tr} (ℓ_1)	n_{conv} (ℓ_1)	$\hat{n}-n$ (ℓ_1)	n_{tr} (ℓ_2)	n_{conv} (ℓ_2)	$\hat{n}-n$ (ℓ_2)
1: EA	–	301	300	–	100	99
1: LA	104	464	4	103	208	3
2: EA	–	30	29	–	40	39
2: LA	102	459	2	103	174	3

TABLE III
Performances of Detection for Replay Attack (Digit Based)

Attack Dataset	n_{tr} (ℓ_1)	n_{conv} (ℓ_1)	$\hat{n}-n$ (ℓ_1)	n_{tr} (ℓ_2)	n_{conv} (ℓ_2)	$\hat{n}-n$ (ℓ_2)
1: EA	–	535	534	–	535	534
1: LA	118	505	18	118	203	18
2: EA	–	424	423	–	344	343
2: LA	114	481	14	114	404	14

Next, we calculate the "delay in detection," i.e., the difference between the estimated versus actual attack point, $(\widehat{n}-n)$, using ℓ_1 and ℓ_2 norms and report their values for the encoding as well as digit/component-based approaches. Regarding the encoding-based algorithm, the $(\widehat{n}-n)$ values for the replay and bias-injection attacks are presented in Tables I and II, respectively. For the digit/component-based algorithm, we show the $(\widehat{n}-n)$ values for the replay and bias-injection attacks in Tables III and IV, respectively.

The fact that Table I (resp., Table II) is better than Table III (resp., Table IV) suggests that the encoding-based approach

TABLE IV
PERFORMANCES OF DETECTION FOR BIAS-INJECTION ATTACK (DIGIT BASED)

Attack Dataset	n_{tr} (ℓ_1)	n_{conv} (ℓ_1)	$\hat{n}-n$ (ℓ_1)	n_{tr} (ℓ_2)	n_{conv} (ℓ_2)	$\hat{n}-n$ (ℓ_2)
1: EA	–	148	147	–	350	349
1: LA	102	334	2	102	244	2
2: EA	–	200	199	–	235	234
2: LA	101	358	1	101	344	1

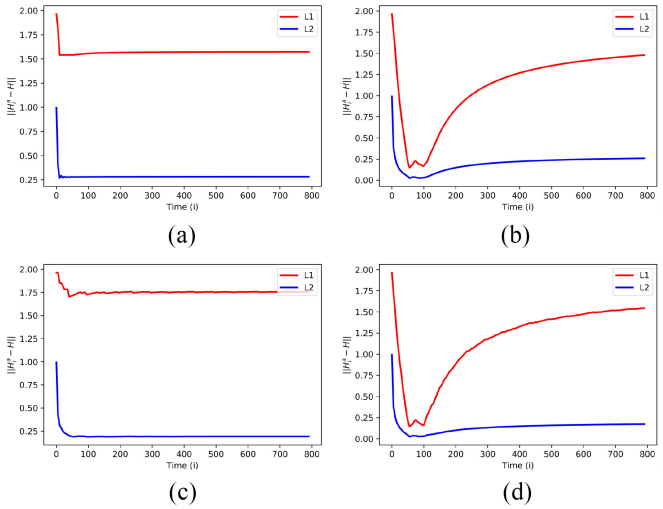

Fig. 4. Plot of d_i for Dataset1 using ℓ_1 versus ℓ_2 norms (red versus blue plots). For the early versus late attacks, the attack start time n is set to 1 versus 100, respectively. As shown in Fig. 4(a) and (c), d_i quickly attains convergence in case of an early attack. For the late attack cases, a trend reversal in the evolution patterns of d_i prior to convergence is evident from Fig. 4(b) and (d). (a) Replay: Early attack. (b) Replay: Late attack. (c) Bias-injection: Early attack. (d) Bias-injection: Late attack.

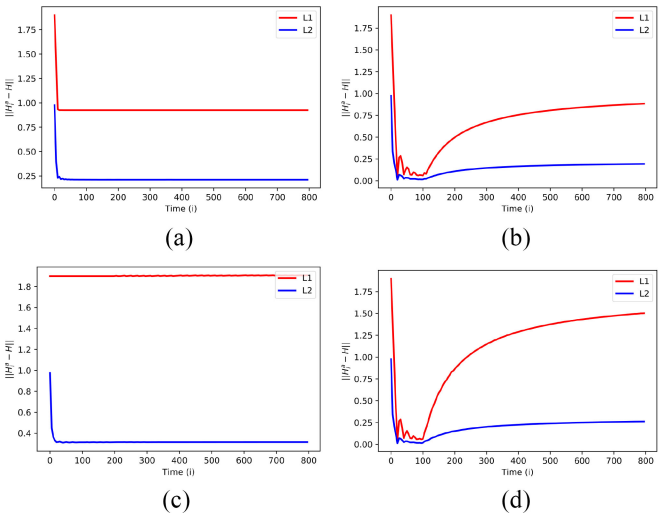

Fig. 5. Plot of d_i for Dataset2 using ℓ_1 versus ℓ_2 norms (red versus blue plots). For the early versus late attacks, the attack start time n is set to 1 versus 100, respectively. As shown in Figs. 5(a) and (c), d_i quickly attains convergence in case of an early attack. For the late attack cases, a trend reversal in the evolution patterns of d_i prior to convergence is evident from Figs. 5(b) and (d). (a) Replay: Early attack. (b) Replay: Late attack. (c) Bias-injection: Early attack. (d) Bias-injection: Late attack.

is superior to the single digit/component-based approach. Limiting the attention to Tables I and II (corresponding to of superior schemes), it can be noted that the delay of detection for the case of late attacks in Tables I and II is comparable. Also, the case of early attack takes longer time compared to the late attacks as expected. This simply means that if an attack occurs even before the detector has had time to converge toward the nominal distribution, then it takes longer for the attack to show up in the histogram measure, as expected. Finally, the time taken to detect an early attack in Table II is higher compared to that in Table I, implying that the bias-injection attack can take longer to detect than the replay attack, in case those get launched early in the detection process (prior to the time the detector has converged to the nominal distribution).

We observe that for the late attack cases, in both the encoding and digit/component-based algorithms, the detection takes the same time with the implementation of ℓ_2 norm as that of ℓ_1 norm. Then, we observe that the performances of both the norms are similar for early attack detection cases. So, we hypothesize that the choice of norm does not significantly affect the delay in detection value for any case.

Next, we compare between the performance of the encoding-based algorithm with that of the digit/component-based algorithm. For replay attacks, the detection by the encoding-based algorithm is significantly faster as compared to its counterpart. Finally, the bias-injection attack results conform to the fact that encoding-based approach is superior in performance to the digit/component-based one. This validates the fact that while detecting an anomaly, the overall change in the data packets is more informative than the change in the individual digits/components present in it.

Finally, we conclude that the reported low $(\widehat{n}-n)$ values for all of the considered cases signify that our algorithm is quite efficient for online detection of attacks in cyber-physical systems.

VII. CONCLUSION

This article presented a *model-free linear time complexity* (in the dimension of data and algorithm parameters) rapid and *online* algorithm for detecting attack-induced anomalies in CPSs by tracking the probability distribution of system data packets in a recursive online manner. The formulation developed a general, statistics-based, data-driven attack detection framework and demonstrated using two common attack cases, namely, replay and bias-injection attacks, which are persistent enough to show up in the form of an altered distribution of the data packets. The formulation involved further identifying two subcases, namely, early and late attacks based on the attack initiation time. Robust notions of convergence and trend reversal were introduced and utilized. It was formally proved that the statistics being tracked can be updated online in linear time, and is guaranteed to converge in finite time, proving the termination of the detection algorithm. The algorithm is implemented and applied to two real-world SCADA datasets from iTrust Lab of a water distribution CPS. The attacks were detected in all 12 cases that we analyzed, and under both approaches (one involving dimensionality reduction and the

other using an individual most sensitive component of the vector data). The estimates of the attack time points are reported, which matched closely with the ground truth. The low delay in detection values substantiates that our algorithm is effective, besides being efficient, for online anomaly detection. Not only can our approach detect quasistationary attacks, it can also detect dynamic attacks as long as there is a difference between the normal versus underattack distributions. Future work can explore the advantages of combining model-based approaches with the proposed histogram-based approach.

ACKNOWLEDGMENT

The authors would like to thank Prof. Subhash Kak for the discussion related to Benford's Law, and iTrust Lab for providing access to the SCADA Data.

REFERENCES

[1] X. Yu and Y. Xue, "Smart grids: A cyber–Physical systems perspective," *Proc. IEEE*, vol. 104, no. 5, pp. 1058–1070, May 2016.
[2] N. Dey, A. S. Ashour, F. Shi, S. J. Fong, and J. M. R. Tavares, "Medical cyber-physical systems: A survey," *J. Med. Syst.*, vol. 42, no. 4, p. 74, 2018.
[3] S. Jianjun, W. Xu, G. Jizhen, and C. Yangzhou, "The analysis of traffic control cyber-physical systems," *Procedia-Soc. Behav. Sci.*, vol. 96, pp. 2487–2496, Nov. 2013.
[4] S. Talukder, M. Ibrahim, and R. Kumar, "Resilience indices for power/cyberphysical systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 4, pp. 2159–2172, Apr. 2021.
[5] Y. Mo, R. Chabukswar, and B. Sinopoli, "Detecting integrity attacks on SCADA systems," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 4, pp. 1396–1407, Jul. 2013.
[6] L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han, "Detecting false data injection attacks on power grid by sparse optimization," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 612–621, Mar. 2014.
[7] D. Ye, T.-Y. Zhang, and G. Guo, "Stochastic coding detection scheme in cyber-physical systems against replay attack," *Inf. Sci.*, vol. 481, pp. 432–444, May 2019.
[8] A. Ghosh, S. Qin, J. Lee, and G.-N. Wang, "FBMTP: An automated fault and behavioral anomaly detection and isolation tool for plc-controlled manufacturing systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 12, pp. 3397–3417, Dec. 2017.
[9] C. Zhou *et al.*, "Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 10, pp. 1345–1360, Oct. 2015.
[10] P. M. Nasr and A. Y. Varjani, "Alarm based anomaly detection of insider attacks in SCADA system," in *Proc. Smart Grid Conf. (SGC)*, 2014, pp. 1–6.
[11] M. Wu, Z. Song, and Y. B. Moon, "Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods," *J. Intell. Manuf.*, vol. 30, no. 3, pp. 1111–1123, 2019.
[12] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for IoT time series," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 112–122, Jan. 2022.
[13] V. K. Singh and M. Govindarasu, "A cyber-physical anomaly detection for wide-area protection using machine learning," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3514–3526, Jul. 2021.
[14] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, no. 4, pp. 1–31, 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0152173
[15] A. Kind, M. P. Stoecklin, and X. Dimitropoulos, "Histogram-based traffic anomaly detection," *IEEE Trans. Netw. Service Manag.*, vol. 6, no. 2, pp. 110–121, Jun. 2009.
[16] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *Proc. KI-2012: Poster Demo Track*, 2012, pp. 59–63.
[17] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, "Decision tree and SVM-based data analytics for theft detection in smart grid," *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1005–1016, Jun. 2016.
[18] "iTrust: Secure Water Treatment Tesetbed," 2018. [Online]. Available: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/ (accessed Sep. 4, 2018).
[19] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Proc. 47th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2009, pp. 911–918.
[20] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, Jan. 2015.
[21] E. Kontouras, A. Tzes, and L. Dritsas, "Impact analysis of a bias injection cyber-attack on a power plant," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11094–11099, 2017.
[22] M. Mohammadpourfard, A. Sami, and Y. Weng, "Identification of false data injection attacks with considering the impact of wind generation and topology reconfigurations," *IEEE Trans. Sustain. Energy*, vol. 9, no. 3, pp. 1349–1364, Jul. 2018.
[23] G. F. Lawler, *Introduction to Stochastic Processes*. Boca Raton, FL, USA: CRC, 2018.
[24] S.-H. Cha and S. N. Srihari, "On measuring the distance between histograms," *Pattern Recognit.*, vol. 35, no. 6, pp. 1355–1370, 2002.
[25] P. K. Pearson, "Fast hashing of variable-length text strings," *Commun. ACM*, vol. 33, no. 6, pp. 677–680, 1990.
[26] P. Perkins and S. Heber, "Identification of ribosome pause sites using a z-score based peak detection algorithm," in *Proc. IEEE 8th Int. Conf. Comput. Adv. Bio Med. Sci. (ICCABS)*, 2018, pp. 1–6.
[27] F. Benford, "The law of anomalous numbers," *Proc. Amer. Philos. Soc.*, vol. 78, no. 4, pp. 551–572, 1938.
[28] W. Hurlimann, "Benford's law from 1881 to 2006," 2006, *arXiv:math/0607168*.
[29] Siemens, Berlin, Germany, "Simatic S7-PLCSIM V5.4 SP8," 2019. [Online]. Available: http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/ (accessed Oct. 22, 2019).
[30] A. T. Al Ghazo, M. Ibrahim, H. Ren, and R. Kumar, "A2G2V: Automatic attack graph generation and visualization and its applications to computer and SCADA networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3488–3498, Oct. 2020.

Ratnesh Kumar (Fellow, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Kanpur, Kanpur, India, in 1987, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 1989 and 1991, respectively.

He is a Palmer Professor of Electrical and Computer Engineering with Iowa State University, Ames, IA, USA, where he directs the Embedded Software, Sensors, Networks, Cyberphysical, and Energy Laboratory. Previously, he held a faulty position with the University of Kentucky, Lexington, KY, USA, and a various visiting positions with the University of Maryland, College Park, MD, USA; Applied Research Laboratory, Pennsylvania State University, State College, PA, USA; NASA Ames, Mountain View, CA, USA; Idaho National Laboratory, Idaho Falls, ID, USA; United Technologies Research Center, East Hartford, CT, USA; and Air Force Research Laboratory, Wright-Patterson AFB, OH, USA.

Prof. Kumar has been a recipient of the D.R. Boylan Eminent Faculty Award for Research from Iowa State University, the Gold Medals for the Best EE Undergraduate, the Best EE Project, the Best All Rounder from IIT Kanpur, the Best Dissertation Award from UT Austin, the Best Paper Award from the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and a keynote speaker and the paper awards recipient from multiple conferences. He was a Distinguished Lecturer of the IEEE Control Systems Society. He has been an Editor of several journals, including IEEE, SIAM, ACM, Springer, IET, and MDPI. He is a Fellow of AAAS.

Ramij Raja Hossain (Graduate Student Member, IEEE) received the bachelor's degree in electrical engineering from Jadavpur University, Kolkata, India, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA.

He served as a Senior Engineer, Distribution with CESC Limited, Kolkata, from 2013 to 2018. His current research includes data-driven MPC, learning accelerated MPC, distributed optimization, game theory, and artificial intelligence-based approaches for security, stability, and control of large-scale cyber-physical systems.

Soumyabrata Talukder (Graduate Student Member, IEEE) received the bachelor's degree in electrical engineering from Jadavpur University, Kolkata, India, in 2009. He is currently pursuing the Ph.D. degree in electrical engineering with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA.

He served as a Project Engineer with GE Grid Solutions (formerly, Alstom Grid), New Delhi, India, from 2009 to 2012, and thereafter, as an Assistant Manager with Central Projects Organization, ITC Ltd., Kolkata, from 2012 to 2017. His current research interests include resilience of complex systems, polynomial optimization, machine learning, stability guaranteed reinforcement learning, and coalition games with power systems as the application domain.

Amit Jena (Student Member, IEEE) received the Integrated M.Sc. degree in mathematics from the National Institute of Technology Rourkela, Rourkela, India, in 2016, and the M.Eng. degree in electrical engineering from Iowa State University, Ames, IA, USA, in 2020.

Alaa T. Al Ghazo (Member, IEEE) received the B.S. degree in electronics engineering from Yarmouk University, Irbid, Jordan, in 2009, the M.S. degree in automation and control engineering from Edinburgh Napier University, Edinburgh, U.K., in 2013, and the Ph.D. degree in electrical engineering and computer engineer from Iowa State University, Ames, IA, USA, in 2020.

He was an Electrical Engineer with LG Electronics, Amman, Jordan, from 2009 to 2010, and with Zoofi Tech, Al Khobar, Saudi Arabia, from 2010 to 2012. He was a Senior Automation Engineer with Reborn Industries, Amman, from 2014 to 2015. He was an Assistant Professor with the Electrical Engineering Department, University of Hartford, Hartford, CT, USA. From 2019 and 2021, he is currently an Assistant Professor with the Department of Mechatronics Engineering, The Hashemite University, Zarqa, Jordan. His research interests include cyber-physical systems (CPSs) and CPS cybersecurity, automation and control systems, artificial intelligence, and machine learning.