Securing Industrial Production from Sophisticated Cyberattacks

Andrew Sundstrom[a], Damas Limoge[b], Vadim Pinskiy[c] and Matthew Putman[d]
Nanotronics Imaging, Brooklyn, NY, U.S.A.

Keywords: Cyberattack, Malicious Attack, Man-in-the-Middle, Stuxnet, Statistical Process Control, Machine Learning, Artificial Intelligence, Innovation Error, Deep Reinforcement Learning.

Abstract: Sophisticated industrial cyberattacks focus on machine level operating systems to introduce process variations that are undetected by conventional process control, but over time, are detrimental to the system. We propose a novel approach to industrial security, by treating suspect malicious activity as a process variation and correcting for it by actively tuning the operating parameters of the system. As threats to industrial systems increase in number and sophistication, conventional security methods need to be overlaid with advances in process control to reinforce the system as a whole.

1 INTRODUCTION

The past 30 years of cyberattacks have witnessed a startling degree of proliferation, adaptation, specificity, and sophistication[1] (see Figure 1). Industrial and military security is the study of walls, physical and digital, which limit malicious insertion or removal of information. For high-security factories and military installations, this means creating systems that are removed from the global computer network and often removed from internal networks. Minuteman ICBM silos, for example, are entirely isolated systems whose launch protocols are seldom updated and whose launch directives are delivered over the Strategic Automated Command and Control System[2]—both protocol and directive systems have relied[3,4], until recently[5,6], on data stored on 8-inch floppy disks, employing an effective combination of network isolation, obsolete IBM Series/1 computing hardware, and low-capacity digital media that is too small for a sophisticated modern cyberattack with a large code footprint. Factory systems (e.g. PLC (Laughton and Warne, 2003) and SCADA (Boyer, 2009)) operate on static code that has been validated and is assumed to be immutable, if not by foreign manipulation. In this study, we focus on an industrial setting to describe our approach.

Although computer methods can be used for validation and security verification prior to deployment, the actual evidence of malicious code installation comes from in-field testing of the entire production line. If malicious code is installed, conventional theory says its effects should manifest themselves in the overall yield of the production line. This method of statistical detection is typically successful in catching direct cyberattacks on single nodes, but it fails against more sophisticated, systemic cyberattacks. The best known example of the new breed of sophisticated cyberattacks[7,8] is Stuxnet[9] (Karnouskos, 2011), which surfaced in 2010[10]. The effects on the centrifuges infected by Stuxnet were not statistically significant with respect to expected baseline behavior, and so each piece of hardware passed nominal Statistical Process Control (SPC) standards. Even if one machine—a single node—starts to behave atypically,

[a] https://orcid.org/0000-0002-3378-4513
[b] https://orcid.org/0000-0002-4963-0500
[c] https://orcid.org/0000-0002-5899-1172
[d] https://orcid.org/0000-0002-1045-1441

[1] https://bit.ly/36BjOkt
[2] https://bit.ly/3041w91
[3] https://bit.ly/2tDxtZE
[4] https://wapo.st/2FvgNGp
[5] https://bit.ly/2QXUJJR
[6] https://nyti.ms/2QAojWP
[7] https://bit.ly/39P5gQ9
[8] https://bit.ly/2s6iaIt
[9] https://bit.ly/36zw056
[10] For a detailed account, refer to the Repository of Industrial Security Incidents (RISI) database, which records "incidents of a cyber security nature that have (or could have) affected process control, industrial automation or Supervisory Control and Data Acquisition (SCADA) systems." https://www.risidata.com/

Sundstrom, A., Limoge, D., Pinskiy, V. and Putman, M.
Securing Industrial Production from Sophisticated Cyberattacks.
DOI: 10.5220/0009148206630670
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 663-670
ISBN: 978-989-758-399-5; ISSN: 2184-4356
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

the effects would manifest in the outputs of that node, measured against historical averages, and the machine would be immediately taken offline, before the entire process can be affected or before other machines can be similarly infected.

In sophisticated modern cyberattacks, however, SPC is ill-suited to detect subtle changes in the operation of many system elements, which individually have small changes, and are thus undetected, but when integrated over time, have major and often catastrophic effects on the entire system. To counter this vulnerability, we propose several methods for detection or correction, based on learning distributions of the system and identifying anomalous behavior. In the corrective case, an agent continuously runs the feedback and feed-forward controls of the system, simultaneously correcting for nominal process variations (e.g. those that occur as a result of process or raw material quality fluctuations) and preventing malicious cyberattacks.

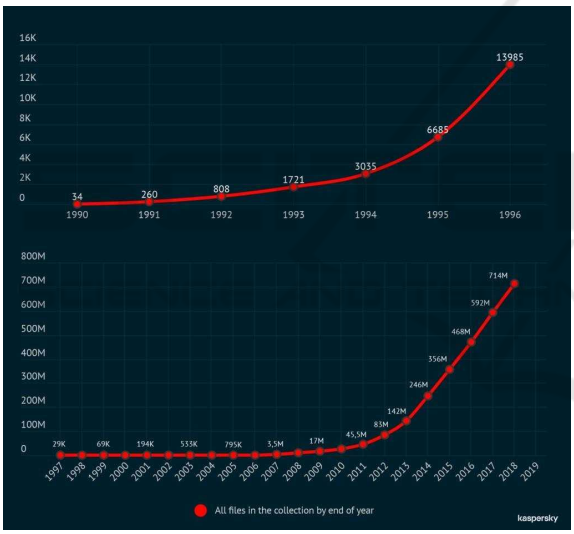

Figure 1: The growth in the number of distinct cyberattacks in the past 30 years, as depicted by Kaspersky Labs.

2 STATISTICAL PROCESS CONTROL

Statistical Process Control (SPC), as popularized by William Edwards Deming in post-war Japan (Deming and Renmei, 1951), (Deming, 1986), (Denton, 1991), (Delsanter, 1992), calls for process standards to be established for each step in the manufacturing process and monitored throughout the production life cycle. The goal is to continuously improve the process through the life cycle.

It is assumed that as long as each node is operating within specification, the final product will also be within specification. The specifications are set based on subject matter expertise and historical performance. The dependability and impact of one node onto the next or subsequent nodes is not directly adjusted in SPC, but rather, each sub-process is examined as an independent entity. This leads to wider margins for the operating condition of each node, preventing the system from ever operating in the absolute highest efficiency or stability.

From a security perspective, this margin can be targeted by sophisticated process cyberattacks. If a single node or several nodes in a system start to operate at the upper bounds (or lower bounds) of their specification, individual alarms will not be triggered, but the overall process quality will be affected. This especially holds for man-in-the-middle cyberattacks, where reported sensor signals, for example, are faked by the malicious code. The life cycle of the node will also be affected, requiring increased downtime for repair. Several layers of downstream nodes will also be affected and over time, the continual drift of the system will tend toward non-compliance. By that point, the correction needed to recover the system would be massive and cost-prohibitive.

3 A MATHEMATICAL MODEL OF PROCESS CONTROL DISRUPTION

A factory can be defined using to a wide variety of topological schemes, including feedback and feed-forward organization. Here we give a simple model of a factory, offering just enough complexity to facilitate a rigorous presentation of the approaches outlined below, which can operate on system topologies of arbitrary complexity.

Accordingly, we define a factory, F, as a strictly linear sequence of n processing nodes, labeled $1,\ldots,n$, connected in a forward-linked chain.

$$F :\rightarrow 1 \rightarrow 2 \rightarrow \cdots \rightarrow i \rightarrow \cdots \rightarrow n$$

The processing done by each node i has two attribute distributions, an expected distribution, Q_i, and an observed distribution, P_i. Q_i is characterized by μ_{Q_i} and σ_{Q_i}. If $Q_i = N(\mu_{Q_i}, \sigma^2_{Q_i})$, then Q_i is completely characterized. P_i is characterized by μ_{P_i} and σ_{P_i}. If $P_i = N(\mu_{P_i}, \sigma^2_{P_i})$, then P_i is completely characterized.

We define the *damage* caused by node i to be the Kullback–Leibler divergence (Kullback and Leibler, 1951), (Kullback, 1959) of P_i with respect to Q_i:

$$d_i = D_{KL}(P_i||Q_i) = \sum_{x\in\chi} P_i(x) log\left(\frac{P_i(x)}{Q_i(x)}\right). \quad (1)$$

For this simple, illustrative model, we assume damage is cumulative, more specifically additive, across F, and so

$$d_F = \sum_{i=1}^{n} d_i. \tag{2}$$

Consider the *Statistical Process Control* (SPC) protocol, which uses μ_i and σ_i to determine if processing at node i is in or out of control on the basis of whether $x \in P_i$ falls within $\mu_{Q_i} \pm 3\sigma_{Q_i}$.

Now consider two adversarial cases for node i, where $Q_i = N(\mu_{Q_i}, \sigma^2_{Q_i})$ and $P_i = N(\mu_{P_i}, \sigma^2_{P_i})$:

- Case 1 (max-burn): $\mu_{P_i} = \mu_{Q_i} + 3\sigma_{Q_i} - \varepsilon_i$ and $\sigma_{P_i} = \frac{\varepsilon_i}{5}$
- Case 2 (min-burn): $\mu_{P_i} = \mu_{Q_i} - 3\sigma_{Q_i} + \varepsilon_i$ and $\sigma_{P_i} = \frac{\varepsilon_i}{5}$

By setting $\sigma_{P_i} = \frac{\varepsilon_i}{5}$, we ensure 99.99943% of the observed events stay in control.

This gives a definition of SPC satisfaction:

$$SPCSAT = \mathbf{1}(\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\{x : x \in P_i|_{\mu_{P_i} \pm 5\sigma_{P_i}} \text{ and } x \notin Q_i|_{\mu_{Q_i} \pm 3\sigma_{Q_i}}\} = \varnothing) > \tau_s), \tag{3}$$

for some ratio $\tau_s \in (0,1]$, typically 1, meaning the processing of all nodes $1, \ldots, n$ is in control.

By definition, Cases 1 and 2 satisfy SPC; their P_i burn at, but are safely contained within, the upper and lower bounds of the Q_i, respectively.

Despite satisfying SPC, Cases 1 and 2 do accumulate measurable damage. This is the essence of how a cyberattack like Stuxnet works; it causes processing nodes to operate within established statistical tolerances, while accumulating damage to the manufactured products.

Consider Case 1. The probability density functions for Q_i and P_i are given by

$$q_i(x) = \frac{1}{\sqrt{2\pi\sigma^2_{Q_i}}} e^{-\frac{(x-\mu_{Q_i})^2}{2\sigma^2_{Q_i}}} \tag{4}$$

and

$$p_i(x) = \frac{1}{\sqrt{2\pi(\frac{\varepsilon_i}{5})^2}} e^{-\frac{(x-(\mu_{Q_i}+3\sigma_{Q_i}-\varepsilon_i))^2}{2(\frac{\varepsilon_i}{5})^2}}, \tag{5}$$

respectively.

Since most of the probability mass of P_i is near $x = \mu_{Q_i} + 3\sigma_{Q_i} - \varepsilon_i$ and is otherwise close to 0 (see Figure 2), we find

$$q_i(x) \approx \frac{1}{\sqrt{2\pi\sigma^2_{Q_i}}} e^{\frac{(3\sigma_{Q_i}-\varepsilon_i)^2}{2\sigma^2_{Q_i}}} \tag{6}$$

and

$$p_i(x) \approx \frac{1}{\sqrt{2\pi(\frac{\varepsilon_i}{5})^2}}. \tag{7}$$

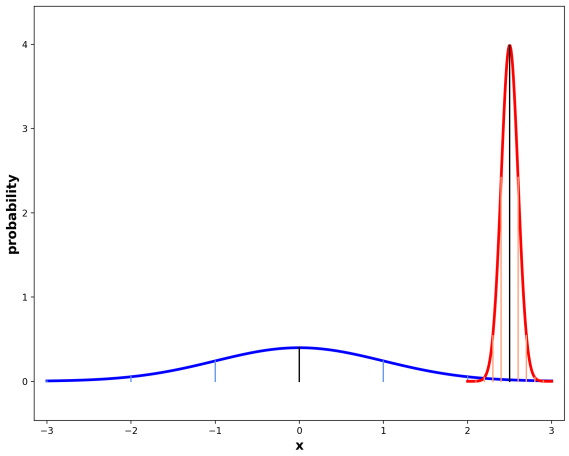

Figure 2: Example expected distribution Q_i (blue) and example observed distribution P_i (red) for a given node i in Case 1: P_i is designed to occupy the rightmost extreme subrange of Q_i, where the probability mass of P_i for $x \in \mu_{P_i} \pm 5\sigma_{P_i}$ is entirely contained within $x \in [\mu_{Q_i} + 3\sigma_{Q_i} - 2\varepsilon_i, \mu_{Q_i} + 3\sigma_{Q_i}]$. Shown here: $\{\mu_{Q_i} = 0, \sigma_{Q_i} = 1\}, \{\varepsilon_i = 0.5, \mu_{P_i} = \mu_{Q_i} + 3\sigma_{Q_i} - \varepsilon_i = 2.5, \sigma_{P_i} = \frac{\varepsilon_i}{5} = 0.1\}$.

Substituting these into the definition of d_i, we derive

$$d_i = \frac{5}{\varepsilon_i\sqrt{2\pi}}\left[log\left(\frac{5\sigma_{Q_i}}{\varepsilon_i}\right) + \frac{(\varepsilon_i - 3\sigma_{Q_i})^2}{2\sigma_{Q_i}}\right]. \tag{8}$$

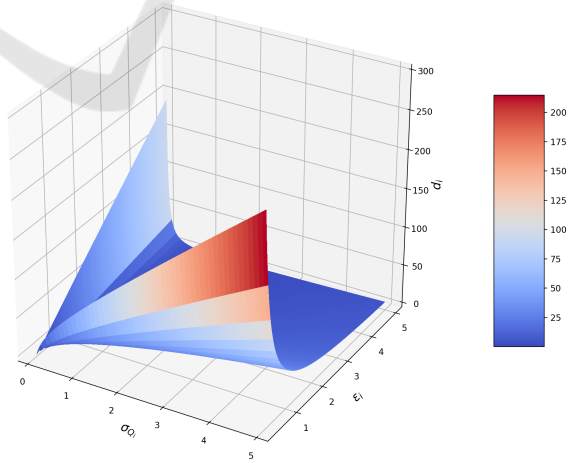

Figure 3: Damage, d_i, as a function of expected standard deviation, σ_{Q_i} and proximity to SPC tolerance, ε_i, for a given node i in Cases 1 and 2, as given by (8).

We depict (8) in Figure 3. Here we see that as expected variance, σ_{Q_i}, shrinks to 0 while proximity to

SPC tolerance, ε_i, grows, then damage, d_i increases exponentially. Similarly, as ε_i shrinks to 0 while σ_{Q_i} grows, then d_i increases exponentially. These two maxima represent two intuitively undesirable situations, neither of which are detected by SPC: $\{\sigma_{Q_i} \to 0, \varepsilon_i \to \infty\} \implies$ the stringency of Q_i outstrips the proximity of P_i to the extrema of Q_i; and $\{\sigma_{Q_i} \to \infty, \varepsilon_i \to 0\} \implies P_i$ occupies an ever-narrowing subrange of Q_i.

Consider Case 2. The only difference from Case 1 is the probability density function for P_i, given by

$$p_i(x) = \frac{1}{\sqrt{2\pi(\frac{\varepsilon_i}{5})^2}} e^{-\frac{(x-(\mu_{Q_i}-3\sigma_{Q_i}+\varepsilon_i))^2}{2(\frac{\varepsilon_i}{5})^2}}. \quad (9)$$

Since most of the probability mass of P_i is near $x = \mu_{Q_i} - 3\sigma_{Q_i} + \varepsilon_i$ and is otherwise close to 0, we find $p_i(x)$ is identical to Case 1, so the remaining derivation follows identically, as one expects from the symmetric normal distribution.

Hence, in Cases 1 and 2, we have an expression that defines a measurable damage for each node i in F despite their having satisfied SPC. Sufficiently large cumulative damage implies process control disruption:

$$d_F > \tau_d. \quad (10)$$

4 FORMULATING THE DAMAGE RECOVERY PROBLEM

SPC is a static, non-interventional approach to process control, where well-defined statistical properties are passively observed to pass or fail at each node, and only after the last node's processing is a decision made as to whether to keep or discard the manufactured product.

By contrast, we consider a dynamic, interventional approach to process control, where each node subsequent to the node causing detected damage is woven into an optimization problem—the damage recovery problem–and actively controlled to instantiate a solution to it. This is done near real-time and while each cycle is ongoing, rather than at the end of a given cycle.

If a control system detects node k has caused damage (i.e., has produced a damaged or distorted distribution), then intuitively, we want to employ a control strategy that samples from P_k, and generates all subsequent resulting distributions flowing from it, $P_{k+1}, \ldots, P_n$, such that the remaining cumulative damage, $d_{k+1} + \cdots + d_n$, is minimized. Accordingly, we formulate the damage recovery problem as

$$\underset{\{P_{k+1},\ldots,P_n\}}{\operatorname{argmin}} \left\{ \sum_{i=k+1}^{n} D_{KL}(P_i||Q_i) \right\}. \quad (11)$$

5 PROPOSED SOLUTION ARCHITECTURES

In the interest of minimizing the damage, D_{KL}, as defined in (11), one might consider simply applying more stringent controls to each node of the process, effectively minimizing the σ_{Q_i}, or attempt to idealize the situation by assuming zero (or nearly zero) observable variance in practice, effectively minimizing the σ_{P_i}. More stringent controls translate into exponential increases in expense in the number of nodes, n, and observing ideally low levels of variance may very well be impossible over long time spans for any realistic manufacturing process.

As a counter to the trivial solution above, several advanced methods for detection or correction are considered. One naturally follows the other, though a correction method may include only implicit detection. Three potential solutions to the correction of damage are outlined below, though their efficacy will be explored later. The first uses adaptive methods to control a system with an unknown disturbance, in the simplest case, a constant disturbance. To generalize the distribution description from (1), a single-input, single-output system is established in state-space form as

$$\begin{aligned} \dot{\vec{x}}_i &= A_i \vec{x}_i + B_i u_{\varepsilon,i}(t) \\ y_i(t) &= C_i^\top \vec{x}_i, \end{aligned} \quad (12)$$

for $\vec{x}_i$ defined as arbitrary states of the system, y defined as the output of the system, and A, B and C are system matrices defining the ordinary differential equation of the underlying dynamics. The input of this system, u_ε, is a noisy input signal defined by

$$u_{\varepsilon,i} = u_i(t) + \varepsilon_t, \quad (13)$$

where ε_t is additive noise contributed by $\varepsilon_t \sim \mathcal{N}(\mu_{\varepsilon,i}, R_i)$ Additionally, the observed output, y_ν, is a function of the system output in (12) as

$$y_{\nu,i} = y_i(t) + \nu_t \quad (14)$$

for a similarly noisy signal measurement, with $\nu_t \sim \mathcal{N}(\mu_{\nu,i}, \sigma^2_{\nu,i})$. This notation is reconciled with that of Sections 3 – 4 by establishing that $y_{\nu,i} \sim Q_i$ for a given node, i, of a process. In an unaffected system, the mean of the noise contributions are zero, such that $\mu_{\varepsilon,i} = \mu_{\nu,i} = 0$. In a malicious cyberattack, however, the deviation manifests as non-zero mean input noise.

5.1 Innovation Error Distribution via Kalman Filter State Estimation

A common approach to estimating states within a system similar to (12) is a Kalman Filter (Kalman, 1960) (KF), which can be extended to nonlinear systems as well. The formulation of the KF is generally reliant on zero mean noise, but in the case of a malicious cyberattack, the offset of the input instruction would manifest as a non-zero mean additive noise. Therefore, a KF can be constructed for the presumed linear time-invariant system described in (12). The filter is constructed using measurements of output, $y_{\nu,i}(t)$ for a node of a process, and the canonical, untouched input instructions $u_i(t)$. If the process is correctly calibrated, the input/output sensor measurements should have zero mean noise, but in the case of a malicious cyberattack there would be a non-zero bias, as depicted in Figure 2. The filter (Thrun et al., 2005) is constructed as

$$\begin{aligned}
\bar{\vec{x}}_{i,k} &= A_i \hat{\vec{x}}_{i,k-1} + B_i u_{i,k} \\
\bar{\Sigma}_{i,k} &= A_i \Sigma_{i,k-1} A_i^\top + R_i \\
K_{i,k} &= \bar{\Sigma}_{i,k} C_i (C_i^\top \bar{\Sigma}_{i,k} C_i + \sigma^2_{\nu,i})^{-1} \\
\hat{\vec{x}}_{i,k} &= \bar{\vec{x}}_{i,k} + K_{i,k}(y_{\nu,i,k} - C_i^\top \bar{\vec{x}}_{i,k}) \\
\Sigma_{i,k} &= (I - K_{i,k} C_i^\top)\bar{\Sigma}_{i,k}
\end{aligned} \tag{15}$$

for the k^{th} sample of a process node, i, where $\bar{\cdot}$ is the measurement update notation, $\Sigma_{i,k}$ is the covariance of the state prediction, R_i is the covariance of the input noise, ε_t, and $K_{i,k}$ are the Kalman gains. With a large enough sample, the innovation distribution, $\tilde{y}_{i,k} = y_{\nu,i,k} - C_i^\top \bar{\vec{x}}_{i,k}$ should be $\tilde{y}_{i,k} \sim \mathcal{N}(\mu_{\tilde{y},i,k} = 0, C_i^\top \Sigma_{i,k|k-1} C_i)$. However, with a malicious cyberattack, $\mu_{\tilde{y},i,k} \neq 0$, but this can occur naturally with minimal samples. Once a sample threshold is met, $k > k_{min}$, an alarm can be established for $\tilde{y}_{i,k} > \gamma_i$, where γ_i can be tuned for a process node. If the innovation error is non-zero and above the threshold γ_i, a malicious cyberattack might be occurring. Figure 4 shows a schematic of the filter described in (15). It should be noted that a twin controller must be maintained and isolated from the primary PLC controllers. This is used as an unbiased reference for the kalman filter. The limitations of this method are requiring a known system, of the form A, B, and C. Additionally, the duplicated controller represent an additional challenge for maintaining software links. To alleviate these issues, agnostic inferential methods can be used as abstract representations of the system node.

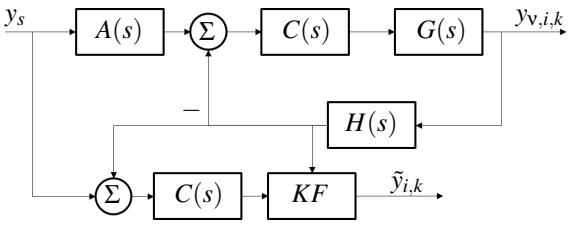

Figure 4: A block diagram of the systems described in (12), incorporating the Kalman filter of (15) for innovation error distribution. The controller, $C(s)$, the plant, $G(s)$, and the measurement, $H(s)$, represent the basic constituents of the nodal control, while the Kalman filter, KF, produced an innovation error. The attack is represented by the block $A(s)$.

5.2 Inferential Methods for Detection and Correction

Artificial Intelligence (AI) (Russell and Norvig, 2010) in the form of Deep Learning (DL) (Goodfellow et al., 2016) has revolutionized image processing (Krizhevsky et al., 2012), machine translation (Johnson et al., 2017), and many other forms of classification (Goodfellow et al., 2014). Specifically, the ability to form complex and non-obvious associations between image pixels and labels, has allowed for higher accuracy of detection than conventional computer vision (Krizhevsky et al., 2012). These methods are also not constrained by the same matching conditions of classical filter methods, such as approach suggested in Section 5.1. This enables more availability for systems that have undetermined or hard to model dynamics.

5.2.1 Autoencoding for Unsupervised Anomaly Detection

Significant attention has been paid to autoencoders trained to detect anomalies (An and Cho, 2015), (Sakurada and Yairi, 2014), (Zhou and Paffenroth, 2017), which, when paired with a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) block in both the encoder and decoder, has been shown to be applicable to anomaly detection. In (Malhotra et al., 2016), an autoencoder was constructed with these LSTM blocks which maintain a state memory of the sequence. For a sequence of measured outputs, $\vec{y}_{\nu,i}$, an unsupervised autoencoder training can be instantiated to map the entropy of output observations on to a parameter set, θ_{AE}, such that

$$\hat{\vec{y}}_{\nu,i} = f(\vec{y}_{\nu,i}, \theta_{AE}). \tag{16}$$

The error of this autoencoder is defined as

$$\tilde{\vec{y}}_{\nu,i} = \vec{y}_{\nu,i} - \hat{\vec{y}}_{\nu,i}, \tag{17}$$

and for a normal operation, $\tilde{\vec{y}}_{\mathrm{v},i} \sim \mathcal{N}(\mu_{\tilde{y},i}, \Sigma_{\tilde{y},i})$, where $\mu_{\tilde{y}}$ and $\Sigma_{\tilde{y},i}$ are fit to the distribution using maximum likelihood. Subsequently, an anomaly score, a_i, for a sequence can be defined as

$$\begin{aligned} a_i &= \mathcal{A}\left(\tilde{\vec{y}}_{\mathrm{v},i}, \mu_{\tilde{y},i}, \Sigma_{\tilde{y},i}\right) \\ &= (\tilde{\vec{y}}_{\mathrm{v},i} - \mu_{\tilde{y},i})^{\top} \Sigma_{\tilde{y},i}^{-1} (\tilde{\vec{y}}_{\mathrm{v},i} - \mu_{\tilde{y},i}). \end{aligned} \tag{18}$$

Similarly to the Kalman Filter formulation in Section 5.1, when the anomaly score, $a_i > \gamma_i$, an anomaly is detected and an alarm is sounded. Figure 5 shows a block diagram of the autoencoder system. The limi-

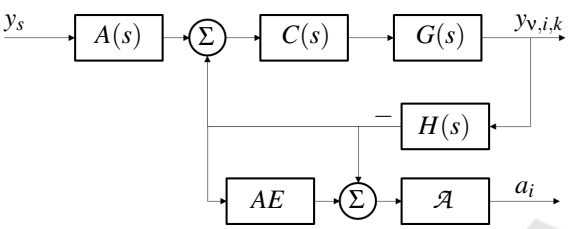

Figure 5: A block diagram of the systems described in (16)–(18), incorporating the autoencoder for anomaly detection. The controller, $C(s)$, the plant, $G(s)$, and the measurement, $H(s)$, represent the basic constituents of the nodal control, while the autoencoder, AE, detects errors, wherein a sufficient anomaly score triggers alarm $\mathcal{A}$. The attack is represented by the block $A(s)$.

tations of the autoencoder approach are a reliance on node-specific training of trajectory prediction models, ignoring the complete factory output. It is possible that while one node is operating appropriately, a previous node has already created damage, and the output of the node in question will already be affected. To correct for this, a more complete system would consider states of each nodal output as an entire trajectory, and adjust the set-point input of each subsequent node accordingly. Such an approach could be accomplished through online optimization methods, such as reinforcement learning.

5.2.2 Deep Reinforcement Learning

In Sections 5.1 and 5.2.1, malicious cyberattack detection was discussed, but no implicit correction was established. However, the definition of damage, d_i, given in (1) suggests a natural structure to formulate a delayed reward function for a reinforcement learning agent seeking to construct a set of distributions, $P_{k+1}, \ldots, P_n$, to solve the damage recovery problem given in (11), through its actions, $\vec{\alpha}_i^{(j)}$, for $i = k+1, \ldots, n$, over some set of iterations, $j = 1, \ldots, m$:

$$R(\vec{\alpha}^{(j)}) = \sum_{i=k+1}^{n} r_i(\vec{\alpha}_i^{(j)}), \tag{19}$$

for

$$r_i(\vec{\alpha}_i^{(j)}) = P_i(\vec{\alpha}_i^{(j)}) log\left(\frac{P_i(\vec{\alpha}_i^{(j)})}{Q_i(\vec{\alpha}_i^{(j)})}\right) \tag{20}$$

In (Lillicrap et al., 2015a), an agent is trained in an actor-critic modality, such that one network produces an action, $\alpha_{i,k}$, given a state, $\vec{x}_{i,k}$ for the k^{th} sample of the i^{th} node of a process, and another network makes a prediction of Q-value, $Q^{\pi}_{i,k}(\vec{x}_{i,k}, \alpha_{i,k} | \theta_{Q,i})$, learned over parameters $\theta_{Q,i}$, where $\pi_i(\vec{x}_{i,k}, \theta_{\pi,i})$ is a learned policy over parameters $\theta_{\pi,i}$. The reward is calculated using a Bellman formulation such that

$$Q_i(\vec{x}_{i,k}, \alpha_{i,k}) = r_{i,k} + \gamma_i Q_i(\vec{x}_{i,k+1}, \pi_i(\vec{x}_{i,k+1}) | \theta_{\pi,i}). \tag{21}$$

Most reinforcement learning agents use an update law corresponding to maximizing the expected return. However, given the formulation in (11), our update law will be

$$\begin{aligned} &-\nabla_{\theta_{\pi,i}} J = \\ &-\mathbb{E}_{\alpha_{i,k} \sim \rho}\left[\nabla_{\alpha_i} Q_i(\vec{x}_{i,k}, \alpha_{i,k} | \theta_{Q,i}) \nabla_{\theta_\pi} \pi_i(\vec{x}_{i,k} | \theta_{\pi,i})\right]. \end{aligned} \tag{22}$$

This update law will minimize the Q-value, thereby minimizing damage, and will manifest in actions aimed at returning the distribution to its canonical shape. One formulation of action could be

$$u_{i,k} = \alpha_{i,k} u^*_{i,k}, \tag{23}$$

where $u_{i,k}$ is the input of (12), $\alpha_{i,k}$ is an instruction modifier, and $u^*_{i,k}$ is the instruction read for a particular sample, k, of node i. If this instruction is corrupted, and that corruption manifests in the states, the policy, $\pi_{i,k}$, will act to correct it. Figure 6 shows a scheme to operate the agent within a factory setting, acting as an outer-loop for the control set-point. An alternative version would directly vary the gain of the control effort from the $C_i(s)$ controllers, subverting the set-point input.

Reinforcement learning has proven effective in complex environments (Sutton and Barto, 2018), (Lillicrap et al., 2015b), (Schulman et al., 2017), (Fujimoto et al., 2018), but requires significant samples to converge. It is recommended that an agent is tuned on simulation data first, then subsequently refined on observed data, before it is deployed in a live system.

Moreover, this approach offers a new way to address system security by bundling process-based malicious cyberattacks into nominal process variations and offers direct control and correction for those variations. The approaches is not simply a method of detection or passive prevention; rather, a cyberattack is assumed to manifest as a routine (e.g. probable) system variation, such as a machine tuning out of norm or a raw material stock moving out of tight specification.

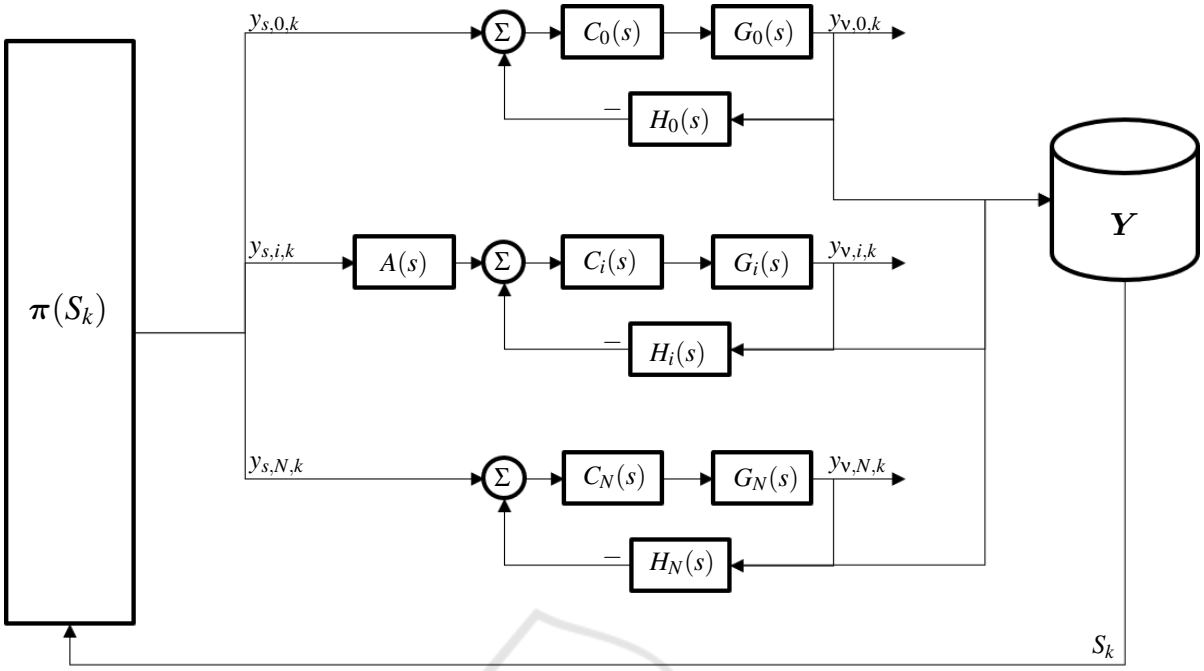

Figure 6: A block diagram of a potential scheme for reinforcement learning based manufacturing control of multiple nodes, $i = 0, \ldots, N$. For each node i, the controller, $C_i(s)$, the plant, $G_i(s)$, and the measurement, $H_i(s)$, represent the basic constituents of the nodal control. Together, the nodes are embedded in a policy-learning feedback loop governed by the state of the system at time k, S_k, sampled from data store Y, and the policy taking the current state as input, $\pi(S_k)$. The attack is represented for a single node, i, by the block $A(s)$.

One would not know that the system was infected, but as long as the agent is making active controls, the final product would be unchanged and the effect of the cyberattack nullified. This also creates latency for nominal IT process to take place and detect intrusion. By focusing on active process control, rather than reactive models, this approach takes advantage of time series data throughout the manufacturing process and not just a single data point for each finished good.

6 CONCLUSION

Physical and digital detachment, which has largely shielded industrial equipment from immediately catastrophic malicious cyberattacks, is no longer sufficient for cyber-defense. In order to enable distributed manufacturing measurement, analysis and feedback, process nodes must be networked, presenting a cyberattack vector for nefarious actors. Even with sophisticated firewalls and real-time alarms to detect corrupted process trajectories, an opportunity persists for minimally invasive alterations to process instructions to operate within acceptable statistical tolerances while damaging the final product irreparably. The answer is not to deprecate the functionality of modern factories in the interest of safeguarding their output, nor is it to pour immense capital into process nodes to ensure their operation fits within an impenetrable distribution of operation.

More sophisticated cyberattacks are able to penetrate standalone equipment, inserting process variations and commands, causing subtle, non-trivial errors that are difficult to detect through conventional process control methods. These cyberattacks can be planned months or years before the actual effect on the instrumentation—possibly within the supply chain of the equipment—making detection and prevention difficult to impossible. The effects they introduce are time and process-state integrative over long scales and across multiple nodes, making detection and correction difficult. We present several alternative approaches for a generalized manufacturing setting that offer not only complete standalone operation, but also active correction for all types of process variation. A sophisticated cyberattack on a specific node or a change in the raw material is corrected by actively changing the processing conditions of subsequent nodes. This agent is continually operational, regardless of the cyberattack, and prevents malicious attempts to alter the steady-state processing from having a noticeable effect on the final quality of the system. This approach increases quality and yield of the

system by actively correcting for all types of nominal variations, while offering increased resistance to possible cyberattacks on the process equipment.

REFERENCES

An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1).

Boyer, S. A. (2009). *SCADA: Supervisory Control And Data Acquisition*. International Society of Automation, USA, 4 edition. ISBN 1936007096.

Delsanter, J. (1992). Six sigma. *Managing Service Quality*, 2(4).

Deming, W. E. (1986). *Out of Crisis*. MIT Center for Advanced Engineering Study, Cambridge, MA, USA.

Deming, W. E. and Renmei, N. K. G. (1951). *Elementary Principles of the Statistical Control of Quality; a Series of Lectures*. Tokyo, Nippon Kagaku Gijutsu Remmei, Tokyo, Japan.

Denton, D. (1991). Lessons on competitiveness: Motorola's approach. *Production and Inventory Management Journal*, 32(3).

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. ISBN 0262035618. http://www.deeplearningbook.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA. MIT Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735—-1780.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Karnouskos, S. (2011). Stuxnet worm impact on industrial cyber-physical system security. In *37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia*. http://papers.duckdns.org/files/2011_IECON_stuxnet.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc. https://bit.ly/39PWOAb.

Kullback, S. (1959). *Information Theory and Statistics*. John Wiley & Sons.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Laughton, M. and Warne, D., editors (2003). *Electrical Engineer's Reference Book*, chapter 16. Newnes, 16 edition. ISBN 0750646373.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015a). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015b). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, USA, 3 edition. ISBN 0136042597. http://aima.cs.berkeley.edu/.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2 edition. ISBN 0262039249. http://incompleteideas.net/book/the-book-2nd.html.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM.