# Intrusion Detection in Digital Twins for Industrial Control Systems

Akbarian, Fatemeh; Fitzgerald, Emma; Kihl, Maria

*Published in:*
2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)

2020

Link to publication

*Citation for published version (APA):*
Akbarian, F., Fitzgerald, E., & Kihl, M. (Accepted/In press). Intrusion Detection in Digital Twins for Industrial Control Systems. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*

# Intrusion Detection in Digital Twins for Industrial Control Systems

Fatemeh Akbarian[1], Emma Fitzgerald[1,2], Maria Kihl[1]
[1]*Department of Electrical and Information Technology, Lund University, Sweden*
[2]*Institute of Telecommunications, Warsaw University of Technology, Poland*

***Abstract*—Nowadays, the growth of advanced technologies is paving the way for Industrial Control Systems (ICS) and making them more efficient and smarter. However, this makes ICS more connected to communication networks that provide a potential platform for attackers to intrude into the systems and cause damage and catastrophic consequences. In this paper, we propose implementing digital twins that have been equipped with an intrusion detection algorithm. Our novel algorithm is able to detect attacks in a timely manner and also diagnose the type of attack by classification of different types of attacks. With digital twins, which are a new concept in ICS, we have virtual replicas of physical systems so that they precisely mirror the internal behavior of the physical systems. So by placing the intrusion detection algorithm in digital twins, security tests can be done remotely without risking negative impacts on live systems.**

***Index Terms*—Intrusion detection, Digital twins, Industrial control systems**

## I. Introduction

Today, smart manufacturing has attracted much attention [1], and the growth of Internet-of-things (IoT) and cloud computing are paving the way for the smart factories that will realize Industry 4.0. As part of Industry 4.0, Industrial Control Systems (ICS), which consists of combinations of control components that act together to achieve an industrial objective, are becoming connected and part of the networked systems in the factory. Although, networked ICSs increase the efficiency of the systems, they may jeopardize the system's security at the same time, since they can provide a critical platform through which attackers may be able to intrude into the system.

Some examples of recent cyber attacks that targeted ICSs are the Stuxnet computer worm attacks on Iran's nuclear installation in 2010 [2], BlackEnergy malware attack on the Ukrainian power grid in 2015 [3], HatMan malware attack on critical infrastructure using Schneider Electrics Safety Instrumented Systems in 2017 [4], malware attack on 40 percent of all ICS in energy organizations protected by Kaspersky Lab solutions in 2017 [5], and the cyber attacks on three U.S. natural gas pipeline companies in 2018 [6]. These attacks demonstrate security weaknesses and the necessity for appropriate security measures to protect ICS infrastructure.

According to this significant increase in cyber attacks, much attention has been paid to intrusion detection in ICS over the recent years.

Digital twin is a rather new concept in industry. With digital twins, we have virtual replicas of physical systems so that they precisely mirror the internal behavior of the physical systems [7]. So using this virtual environment for security tests instead of the real system prevents any interference with the live systems.

Hence, in this paper, we propose a digital twin based intrusion detection technique due to a couple of reasons. First, applying security tests may have some negative effects on the live systems and cause to reduce the efficiency of the systems. Also, an intrusion detection system in the digital domain can include methods that require much more computing resources than if deployed in the real system. For instance, machine learning techniques usually are hard to realise in the physical domain, where there often are embedded devices with limited computing power and more restrictive programming models.

There are a few papers that have suggested intrusion detection as a use case for digital twins. To the best of our knowledge, so far, only two papers have been published that show how the concept of digital twins can be used to implement intrusion detection systems. Authors in [8], defined two rules, namely, safety and security rules, that specific digital twins must adhere to. However, this research missed the synchronization between digital twins and the real systems. So real systems' data is not incorporated into the implemented intrusion detection. Moreover, the proposed rules are very limited as an intrusion detection method for detecting cyber attacks. Regarding these issues, authors in [9] proposed a passive state replication to create synchronization between physical systems and digital twins that is a fundamental requirement for realizing the intrusion detection using digital twins. However, this synchronization method only copy some limited data of physical system to digital twin and it does not make the digital twin able to follow the physical system continuously for instance, when there are unexpected changes in the physical system. Further, the authors proposed a behavior-specification-based intrusion detection to determine whether the system's behavior during runtime diverges from the predefined correct behavior due to an intrusion. However,

This paper was supported by the Celtic-Next project 5G PERFECTA funded by Vinnova, and the SSF project SEC4FACTORY under grant no. SSF RIT17-0032. The authors are part of the Excellence Center at Linköping-Lund on Information Technology (ELLIIT) strategic research area, and the Nordic University Hub on Industrial IoT (HI2OT) funded by NordForsk. Maria Kihl is partially funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

creating the specification of the system's correct behavior typically requires processing effort, whereas in this paper the authors sidestep this issue by making the assumption that the specification of the system is readily available.

Motivated by synchronization challenges in digital twins, in our prior work [10] we proposed an architecture to implement a digital twin that makes it able to follow its physical counterpart's behavior continuously and guarantees synchronization between digital twins and physical systems. In this paper, we equip this architecture with a novel intrusion detection algorithm that unlike [9] does not need a specification of the system's correct behavior. We demonstrate how the digital twin concept can be used for intrusion detection. Further, we develop a novel algorithm in the digital twin for timely detection of attacks on ICS. Also, we propose an approach to diagnose the type of attack after detecting it by classification of different types of attacks. Finally, we evaluate the capability of the proposed anomaly detection algorithm through simulation studies.

## II. Targeted System

The targeted system, which concerns industrial control systems, is illustrated in Fig. 1. In this figure, the physical domain shows an industrial system, which is composed of several different control systems. These control systems can represent a factory where there are different kinds of machines and robots. A digital twin for the system is deployed in the cloud using the architecture in our prior work [10]. There will always be a network between the physical domain and the cloud, and also there can be a network between the components of a control system inside the physical domain. These networks create the possibility of cyber attacks on the signals that are sent through them.

In this paper, as Fig. 1 shows, an intrusion detection system is deployed in the digital twin in order to detect cyber attacks on the real system. Also, we consider attacks that intrude into the network and try to manipulate the measurement signal $y$ in order to drive the system to an unsafe state.

We consider two Scaling and Ramp attacks in which the attacker inject false data into the signal and modify it. According to [11], [12] and [13], we can model these attacks as follows:

- Scaling attack: In this type of attack, the measurement signal is manipulated and its value depending on the amount of scaling attack parameter $\lambda_s$, is converted to a value greater or less than the actual value:

$$y^*(t) = \begin{cases} y(t) & \text{for } t \notin \tau_a \\ (1+\lambda_s) \cdot y(t) & \text{for } t \in \tau_a \end{cases} \tag{1}$$

  where $\tau_a$ indicates the attack period.

- Ramp attack: In this type of attack, since the beginning of the attack, $\lambda_r.t$ is added to the actual signal and depending on the amount of $\lambda_r$, increases or decreases the value of the signal:

$$y^*(t) = \begin{cases} y(t) & \text{for } t \notin \tau_a \\ y(t) + \lambda_r.t & \text{for } t \in \tau_a \end{cases} \tag{2}$$

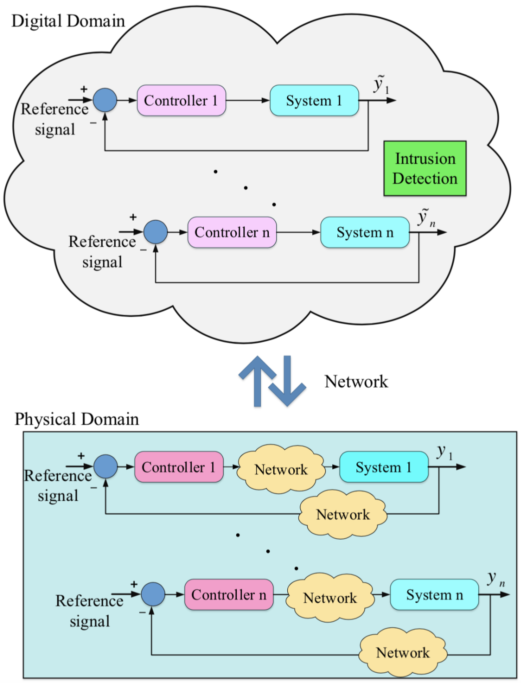

Fig. 1. Targeted system overview.

Since different attacks can have different effects on the system, different mitigation methods may be needed to deal with them. Therefore, after detecting the attack, it is also important to classify the attack in order to choose the best mitigation method for it. Hence, in next section, we explain our proposed method to detect these attacks in a timely manner and classify them.

## III. Proposed Solution

In this paper, we propose a novel intrusion detection, and we also propose to apply this intrusion detection to digital twins instead of the real system. Our proposed solution consists of two parts: attack detection and attack classification.

### A. Attack Detection

In order to detect an attack on the system, we propose to use a Kalman filter [14] to estimate the correct signals in the system. The Kalman filter uses input and output signals of the system to estimate the correct output of the system. This means that the Kalman filter optimally removes the destructive effects of the attack and noises from the manipulated signal and can estimate the correct behavior of the system. This estimated signal can then be used to detect the occurrence of the attack. In order to design a Kalman filter, first, an observable state-space model of the system is needed. The Kalman filter will be placed in the digital twin, and as the digital twin is created based on [10], we can assume that there exists a simulated model of the real system that has been obtained easily by system identification algorithms. Therefore, it is only necessary to create an observable realization of this model to generate an observable state-space model.

After designing the Kalman filter, input and output signals of the system, which, the Kalman filter has been designed for, should be sent to the Kalman filter to estimate the correct output signal.

By considering process noise and measurement noise in the system, our targeted system can be modelled as follows:

$$\begin{array}{ll} x_{\mathrm{k+1}} = Ax_{\mathrm{k}} + Bu_{\mathrm{k}} + Gw_{\mathrm{k}} & w \rightarrow N(0,Q) \\ y_{\mathrm{k}} = Cx_{\mathrm{k}} + Fv_{\mathrm{k}} & v \rightarrow N(0,R) \end{array} \tag{3}$$

In this model $x$ is the state vector, $y$ is the output signal which is measured by sensors, $u$ is the input signal which is generated by the controller, $w$ is process noise, $v$ is measurement noise and subscript $k$ shows time instance. Here, we consider process noise and measurement noise to be white noise with covariances $Q$ and $R$ respectively. Also, $A$, $B$, $C$, $G$, and $F$ are coefficient matrices.

A Kalman filter for this system will be designed using the following recursive algorithm, which consists of two parts: time update and measurement update [14]. The time update part consists of the following steps:

$$1)\quad \hat{x}_{\mathrm{k|k-1}} = A_{\mathrm{k}}\hat{x}_{\mathrm{k-1k-1}} + B_{\mathrm{k}}u_{\mathrm{k}} \tag{4}$$

$$2)\quad P_{\mathrm{k|k-1}} = G_{\mathrm{k-1}}Q_{\mathrm{k-1}}G_{\mathrm{k-1}}^{\mathrm{T}} + A_{\mathrm{k-1}}P_{\mathrm{k-1|k-1}}A_{\mathrm{k-1}}^{\mathrm{T}} \tag{5}$$

and the measurement update part consists of following steps:

$$3)\quad K_{\mathrm{k}} = P_{\mathrm{k|k-1}}C_{\mathrm{k}}^{\mathrm{T}}\left(C_{\mathrm{k}}P_{\mathrm{k|k-1}}C_{\mathrm{k}}^{\mathrm{T}} + F_{\mathrm{k}}R_{\mathrm{k}}F_{\mathrm{k}}^{\mathrm{T}}\right)^{-1} \tag{6}$$

$$4)\quad \hat{x}_{\mathrm{k|k}} = \hat{x}_{\mathrm{k|k-1}} + K_{\mathrm{k}}\left(y_{\mathrm{k}} - C_{\mathrm{k}}\hat{x}_{\mathrm{k|k-1}}\right) \tag{7}$$

$$5)\quad P_{\mathrm{k|k}} = \left(I - \mathrm{K}_{\mathrm{k}}C_{\mathrm{k}}\right)P_{\mathrm{k|k-1}} \tag{8}$$

where $\hat{x}$ is the estimated state vector, $P$ is the estimating covariance matrix and $K$ is the Kalman gain.

However, one problem of a Kalman filter is that it requires some characteristics of the noise. Q and R are the covariance matrices of process and measurement noises, respectively, and these are usually not known. To overcome this problem, inspired by the proposed method in [15], we propose a particle swarm optimization (PSO) and combine this with the Kalman filter. In this method, first, in offline mode using a combination of PSO and the Kalman filter, the optimal values of Q and R can be found. Then, these obtained values are injected into the Kalman filter for estimation in online mode.

Using this Kalman filter, state variables of the system are estimated. The system's output can also be estimated based on these state variables and using the system model as follows:

$$\hat{y}_{\mathrm{k}} = C\hat{x}_{\mathrm{k}} \tag{9}$$

Now by comparing the estimated signal $\widehat{y_k}$ with the output signal $\tilde{y}_k$, which is virtual version of the measurement signal $y$ and regenerated by the digital twin and exactly follows y, the residual signal is generated:

$$r_k = \tilde{y}_k - \widehat{y_k} \tag{10}$$

In order to distinguish attacks from noises and detect the occurrence of an attack, it is necessary to use a detector. Actually, the detector helps to make the effects of attacks and noises on the signal more prominent and filter impacts of noises and prevent false alarms. Our detector uses residual signal $r$ and generates $h$ so as to detect attacks as follows:

$$h_k = \sup_{k-k_0<i<k} |r_i|, \quad \begin{cases} H_0: \text{ if } & h_k \leq \text{ threshold} \\ H_1: \text{ if } & h_k > \text{ threshold} \end{cases} \tag{11}$$

where the hypothesis $H_0$ indicates the normal operation of the system and $H_1$ indicates the abnormal mode of the system.

Since the Kalman filter cannot distinguish the changes caused by the attack from the changes due to the presence of the noise, a threshold for filtering the effects of noise and preventing false alarms should be considered. Based on 68-95-99.7 rule, in a Gaussian distribution, 68.27%, 95.45%, and 99.73% of the values lie within one, two, and three standard deviations of the mean, respectively. Since the noise in this paper is assumed to be Gaussian noise with zero mean, by considering a threshold equal to $3\sigma$, where $\sigma$, is the standard deviation of the measurement noise, 99.73% false alarms that may occur due to this noise can be filtered out.

### *B. Attack Classification*

The goal of the classification is to categorize data into distinct classes. Support Vector Machine (SVM) is a machine learning approach used for classification, and in this approach, a model is generated based on training data and then it can be used to predict the class of new data. SVM supports both binary classification and multi classification. In this paper, we want to classify the state of the system as Normal, Scaling attack or Ramp attack. Therefore, a multi classification must be used. There are several methods for multi-class SVM, but we apply the one-against-one method which is the most efficient one based on [16]. In the one-against-one method, $k(k-1)/2$ classifiers are created: $\left(w^1\right)^T\phi(x) + b^1$ , ... , $\left(w^{k(k-1)/2}\right)^T\phi(x) + b^{k(k-1)/2}$ where $k$ is the number of classes. Each of these classifiers is trained on data from two classes. For training data from the $ith$ and the $jth$ classes, the following binary classification problem is solved:

$$\begin{array}{ll} \min\limits_{w^{ij},b^{ij},\xi^{ij}} & \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum\limits_t \xi_t^{ij} \\ & \left(w^{ij}\right)^T\phi\left(x_t\right) + b^{ij} \geq 1 - \xi_t^{ij}, \text{ if } y_t = i \\ & \left(w^{ij}\right)^T\phi\left(x_t\right) + b^{ij} \leq -1 + \xi_t^{ij}, \text{ if } y_t = j \\ & \xi_t^{ij} \geq 0 \end{array} \tag{12}$$

where $\phi$ is the function that maps training data $x_t$ to a higher dimensional space to make data more separable and $C$ is the penalty parameter. By minimizing $\frac{1}{2}(w^{ij})^T w^{ij}$ we maximize $\frac{2}{\|w^{ij}\|}$, which is the margin between two groups of data. When data are not linear separable, there is a penalty term $C\sum\limits_t \xi_t^{ij}$ which can reduce the number of training errors. Actually SVM searches for a balance between the regularization term $\frac{1}{2}(w^{ij})^T w^{ij}$ and the training errors. After all $k(k-1)/2$ classifiers are constructed, if $\operatorname{sign}\left(\left(w^{ij}\right)^T\phi(x) + b^{ij}\right)$ says $x$ is in the $ith$ class, then the vote for the $ith$ class is added by one. Otherwise, the $jth$ is increased by one. Then, $x$ is predicted as the class with the largest vote, and in case that two classes have identical votes we select the one with the smaller index.

In this paper, we propose using the residual signal as $x_t$ for classification. Our motivation behind this decision is that the residual signal is the result from the comparison between the virtual version of the real output signal and the estimated output. Hence, it shows abnormal behavior of the system caused by the attacks, and based on how this abnormal behavior is for each class, we can train a model for attack classification. By applying Ramp attacks and Scaling attacks and also considering the condition where there is no attack, we can generate training data. Then, by labeling these data as class 0 for Normal condition, class 1 for when there is Scaling attack, and class 2 for when there is Ramp attack, and using one-against-one method, we train a model for classification.

## IV. Experiments

We evaluate our proposed approach in Matlab Simulink [17], and in this section we describe our simulation model and our experiments.

### A. Process

In this paper, we use a ball and beam process. The ball and beam system consists of a long beam which can be tilted by an electric motor together with a ball rolling back and forth on top of the beam. This system is open-loop unstable and without a controller, it will swing to one side or the other, and the ball will fall off the end of the beam.

To stabilize the ball, a control system that measures the position of the ball and adjusts the beam accordingly must be used. Our motivation for choosing the ball and beam system is that it is an intrinsically unstable and time-critical system, and any attack on this system can make it unstable quickly. So, evaluating our proposed method on this system can prove the effectiveness of it.

We simulate the ball and beam process using the standard Lagrangian equations of motion for the ball based on [18]. In our ball and beam system, the length of the beam is 1 meter. Therefore, the allowable position for the ball is between 0 and 1, and if it goes outside this range, it will fall.

### B. Digital Twin

Based on our prior work [10], we create a digital twin for the ball and beam process in Matlab Simulink, which follows its physical counterpart continuously. The network between the physical domain and digital part (cloud), is simulated with TrueTime [19]. We consider the network to be an Ethernet with 2.5% packet loss probability and a 40 ms network delay.

### C. Generating Attack Signals

An attack should have two main features. First, it should cause the system to go to an unsafe state, and second, it should not be easily detectable. Ramp attacks add a ramp signal to the measurement signal and causes a change of the position of the ball. If the slope of this ramp signal is high, it changes the ball's position quickly. However, such an attack will be detected easily. So, the slope should be low and change the ball's position gradually in which case it is difficult to detect.

Therefore, we chose 0.5 meters, which is in the middle of the beam, as a setpoint for the ball's position in the controller, and based on this, choosing $0 < \lambda_r \leq 0.1$ in (2) for the Ramp attack is reasonable, since it will cause the ball to fall off and it will be difficult to detect.

Scaling attacks are not so different in terms of how they gradually can change the ball's position. The only main difference between Scaling attacks with different parameters $\lambda_s$, see (1), is the consequence. If $\lambda_s \geq 15$, it will cause the ball to fall. However, choosing $\lambda_s$ out of this range, only causes the ball's position to change on the beam, but it does not cause it to fall and damage the system. So $\lambda_s \geq 15$ is a reasonable range for this type of attacks.

### D. Evaluation of Attack Detection Method

The performance of the detection algorithm is evaluated by measuring the time it takes to detect an attack and comparing it with the time it takes the attack to drive the system to an unsafe state. Here, for the ball and beam system, we compare the time it takes our detection method to detect an attack with the time that the attack takes to cause the ball to fall off.

For this evaluation, we apply Ramp attacks and Scaling attacks with different parameters chosen from the selected range to the measurement signal in the physical domain $y$. Then, we record the time it takes to detect the attack and the time it takes for the attack to cause the ball to fall, which is the time it takes for the position of the ball to increase more than 1 meter or decrease less than 0 meters, and then we compare these two time.

### E. Evaluation of Attack Classification Method

To classify attacks and diagnose the type of an attack, as it said before, we use residual signal $r$ as training data for obtaining a model using SVM, and this model will classify new residual data as Normal, Scaling attack and Ramp attack.

To generate the training data, first, we run the simulation in normal conditions when there are no attacks several times for 60 s and record the residual signal. Using this signal we create a vector containing residual data related to a normal condition that we label as class 0.

In the next step, we apply Scaling attacks with different $\lambda_s$ to the measurement signal $y$ for 60 s, and for each $\lambda_s$, we record the residual signal. In this way, we create a vector containing residual data related to Scaling attacks that are labelled as class 1.

The reason for applying attacks for 60 s is that we want to diagnose the type of attack as soon as it occurs. So, we need to cover different data related to the beginning of the attack, and therefore 60 s is well enough for this purpose.

In the third step, we apply Ramp attacks with different $\lambda_r$ to the measurement signal $y$ for 60 s and for each $\lambda_r$, we record the residual signal. In this way, we create a vector containing residual data related to Ramp attacks that are labelled as class

2. By using this training data with the SVM algorithm, we obtain a model that should be able to determine the class of new data.

The next step is to test this model using testing data. To generate testing data for normal conditions, we run the simulation without applying any attacks for 50 s and record the residual signal as testing data. Testing data for Scaling attacks are generated by applying Scaling attacks at time 30 s for 20 s with different $\lambda_s$ that are chosen from the range $\lambda_s \geq 15$. Testing data for Ramp attacks are generated by applying Ramp attacks at time 30 s for 20 s with different $\lambda_r$ that are selected from the range $0 < \lambda_r \leq 0.1$.

To evaluate the performance of the classification algorithm, the class (label) of the testing data using the obtained SVM model is determined. The accuracy is then calculated as follows:

$$\text{Accuracy} = \frac{\text{The number of correctly predicted data}}{\text{Total number of testing data}} \times 100\% \tag{13}$$

For each of $\lambda_s$ and $\lambda_r$ value and normal condition, the generation and evaluation of testing data is repeated 15 times and finally, the average accuracy for each of $\lambda_s$ and $\lambda_r$ value and normal condition are calculated. Since the variability of accuracy for each of these conditions is really low and we have a quite narrow confidence interval, 15-time repetition is well enough.

## V. Results

In this section, the results of the evaluation of attack detection and classification algorithms are presented.

Fig. 2 shows the signal $h$ and the ball's position in the presence of Ramp attack with $\lambda_r = 0.04$ which is started at 30 s. In this attack, the attacker gradually changes the ball's position and as it can be seen in Fig. 2, the attack causes the ball to fall off at 42.52 s. However, our attack detection algorithm can detect this attack at 31.29 s, which is well before the ball falls off.

Fig. 3 shows the results for a Ramp attack which is started at 30 s with different values of $\lambda_r$. All chosen $\lambda_r$ are small enough to make the attack difficult to detect. For each $\lambda_r$, the blue line shows the time it takes for the ball to fall off, and the red line shows the time it takes for our detection algorithm to detect the attack. As can be seen in the figure, all attacks can be detected before the attack can drive the system to an unsafe state and cause the ball to fall.

Fig. 4 shows a similar evaluation for the Scaling attack with different $\lambda_s$. As can be seen in the figure, although these attacks can quickly affect the system and move the ball off the beam, our proposed detection technique can detect them in a timely manner and before the ball falls off.

To evaluate the classification method, as is said in the previous section, we apply a Ramp attack at time 30 s for 20 s. So, before 30 s, there are no attacks and the condition should be classified as Normal. After 30 s, the condition should be classified as a Ramp attack. By calculating the accuracy, the algorithm can be evaluated.

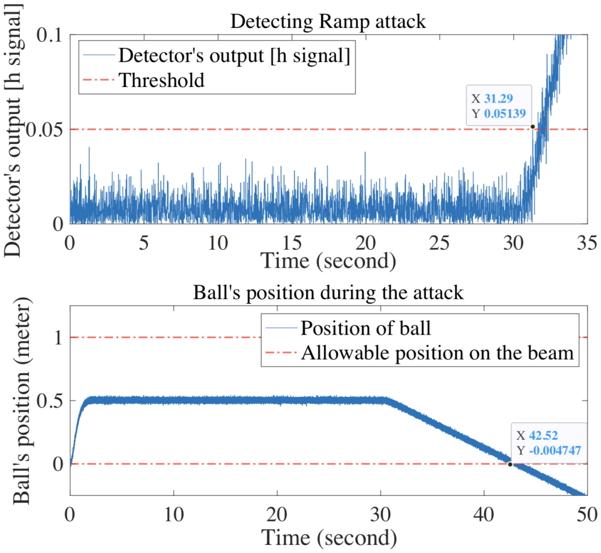

Fig. 2. Detecting Ramp attack with $\lambda_r = 0.04$.

Table I shows the average accuracy for the Ramp attack with different $\lambda_r$. As can be seen, the accuracy for $\lambda_r = 0.03, 0.04, 0.05$ decreases. The reason for this result is that, in this interval, there is an overlap with other classes. However, for other $\lambda_r$ the accuracy is more than 91%. Also the total average accuracy of all cases is 91.27%, which is a high accuracy that shows that our classification method can recognize a Ramp attack.

Similar to the Ramp attack, we apply a Scaling attack to the system at time 30 s for 20 s and calculate the accuracy. Table II shows the average accuracy for the Scaling attack with different $\lambda_s$. The accuracy is more than 98% for all cases, and the total average accuracy is 98.96%, which proves that our classification method can recognize a Scaling attack.

Finally, we evaluate the accuracy when there are no attacks.

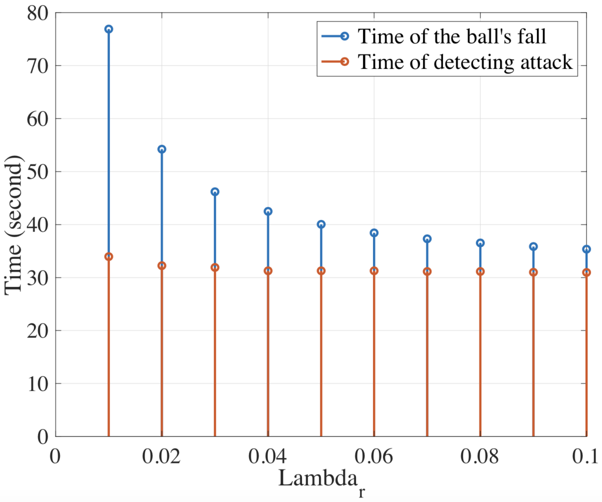

Fig. 3. Detecting Ramp attacks.

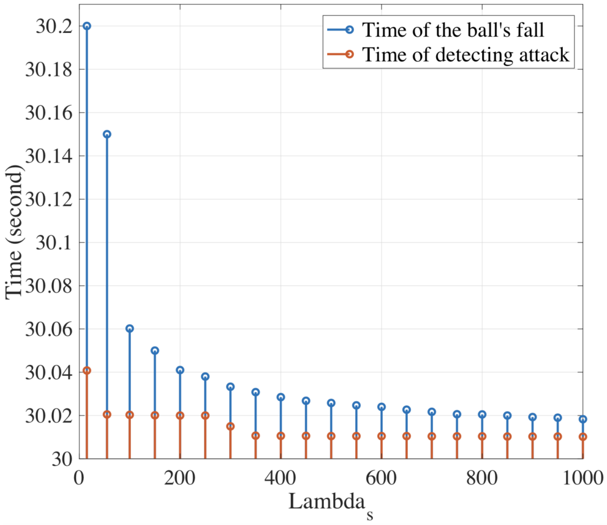

Fig. 4. Detecting Scaling attacks.

In this case, all data should be classified as Normal. The average accuracy for this condition equals 99.94% that shows that our classification method can recognize Normal conditions excellently.

## VI. Conclusions

Industrial control systems are increasingly being connected to communications networks, which make them more vulnerable to cyber attacks. Regarding this issue, in this paper, we propose a digital twin based intrusion detection technique. By deploying the intrusion detection system in the digital domain, more advanced methods that require more computing resources can be developed. Therefore, we have developed and evaluated an intrusion detection mechanism for the digital twin, which can both detect attacks and also classify the type of attack. The intrusion detection mechanism uses a combination of a Kalman filter to detect the attack, a particle swarm optimization algorithm to estimate the noise, and a support vector machine algorithm to classify the attack. Through simulation studies in Matlab Simuling, we show that our detection method is highly effective to detect an attack before the attack can drive the system to an unsafe state. Also, we show that our classification approach can provide a high accuracy when determining the types of attacks. Therefore, our proposed approach can assist industrial control systems with detecting cyber attacks before they cause damage and also classify the type of attack, which will be of great benefit when it comes to choosing the proper mitigation method for each type of attack.

TABLE I
Accuracy of Ramp Attack Classification

| **Parameter $\lambda_r$** | **Accuracy** |
|---|---|
| 0.01 | 91.42% |
| 0.02 | 94.77% |
| 0.03 | 83.97% |
| 0.04 | 87.48% |
| 0.05 | 89.88% |
| 0.06 | 91.37% |
| 0.07 | 92.44% |
| 0.08 | 93.10% |
| 0.09 | 93.85% |
| 0.1 | 94.42% |

TABLE II
Accuracy of Scaling Attack Classification

| **Parameter $\lambda_s$** | **Accuracy** |
|---|---|
| 20 | 98.97% |
| 55 | 99.01% |
| 100 | 99.03% |
| 200 | 98.94% |
| 300 | 98.95% |
| 400 | 98.94% |
| 500 | 98.93% |
| 600 | 98.96% |
| 700 | 98.93% |
| 800 | 99.10% |
| 900 | 98.92% |
| 1000 | 98.95% |

## References

[1] J. Cheng, W. Chen, F. Tao, and C.-L. Lin, “Industrial IoT in 5G environment towards smart manufacturing,” *Journal of Industrial Information Integration*, vol. 10, pp. 10–19, 2018.

[2] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[3] D. Alert, “Cyber-attack against ukrainian critical infrastructure,” *Cybersecurity Infrastruct. Secur. Agency, Washington, DC, USA, Tech. Rep. ICS Alert (IR-ALERT-H-16-056-01)*, 2016.

[4] ICS-CERT, “Hatman—safety system targeted malware,” Mar. 2017. [Online]. Available: https://ics-cert.us-cert.gov/MAR-17-352-01-HatManTargeted-Malware.

[5] Kaspersky Lab ICS-CERT, “Threat landscape for industrial automation systems in h2 2017,” Mar. 2018. [Online]. Available: https://icscert.kaspersky.com/reports/2018/03/26/threat-landscape-for-industrial-automation-systems-in-h2-2017/.

[6] N. S. Malik, R. Collins, and M. Vamburkar, “Cyber-attack, pings data systems of at least four gas networks,” Apr. 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-04-03/day-after-cyberatta ck-a-third-gas-pipeline-data-system-shuts.

[7] M. Farsi, A. Daneshkhah, A. Hosseinian-Far, and H. Jahankhani, *Digital Twin Technologies and Smart Cities*. Springer, 2020.

[8] M. Eckhart and A. Ekelhart, “Towards security-aware virtual environments for digital twins,” in *Proceedings of the 4th ACM workshop on cyber-physical system security*, 2018, pp. 61–72.

[9] ——, “A specification-based state replication approach for digital twins,” in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy*, 2018, pp. 36–47.

[10] F. Akbarian, E. Fitzgerald, and M. Kihl, “Synchronization in digital twins for industrial control systems,” arXiv e-prints, p. arXiv: 2006.03447, June 2020.

[11] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Y. Tsai, and S. Sastry, “Understanding the physical and economic consequences of attacks on control systems,” *International Journal of Critical Infrastructure Protection*, vol. 2, no. 3, pp. 73–83, 2009.

[12] S. Sridhar and M. Govindarasu, “Model-based attack detection and mitigation for automatic generation control,” *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 580–591, 2014.

[13] F. Akbarian, A. Ramezani, M.-T. Hamidi-Beheshti, and V. Haghighat, “Intrusion detection on critical smart grid infrastructure,” in *2018 Smart Grid Conference (SGC)*. IEEE, 2018, pp. 1–6.

[14] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[15] Y. Laamari, K. Chafaa, and B. Athamena, “Particle swarm optimization of an extended kalman filter for speed and rotor flux estimation of an induction motor drive,” *Electrical Engineering*, vol. 97, no. 2, pp. 129–138, 2015.

[16] C.-W. Hsu and C.-J. Lin, “A comparison of methods for multiclass support vector machines,” *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[17] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[18] Ball and Beam: Simulink Modeling. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=BallBeamsection =SimulinkModeling.

[19] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzen, “How does control timing affect performance? analysis and simulation of timing using jitterbug and TrueTime,” *IEEE control systems magazine*, vol. 23, no. 3, pp. 16–30, 2003.