Detect Insider Attacks in Industrial Cyber-Physical Systems Using Multi-Physical Features Based Fingerprinting

ZHEN HONG, Zhejiang University of Technology, China
LINGLING LU*, (corresponding author), Zhejiang University, China
DEHUA ZHENG, Zhejiang Sci-Tech University, China
JIAHUI SUO, Zhejiang University of Technology, China
PENG SUN, Zhejiang University of Technology, China
RAHEEM BEYAH, Georgia Institute of Technology, USA
ZHENYU WEN, Zhejiang University of Technology, China

ICPS software and hardware suffer from low update frequency, making it easier for insiders to bypass external defenses and launch concealed destructive attacks. To address these concerns, we design a device fingerprinting method based on multi-physical features, augmenting current intrusion detection techniques in the ICPS environment. In this paper, we use the sorting system as an example, demonstrating that the proposed device fingerprinting technology has generality in the intrusion detection of ICPS control flow. Specifically, we first formalize the physical model of the sorting system to analyze the critical device features. Then we extract these physical features from the sensor data collected in a physical testbed. Utilizing featurized data, we train a classifier that generates fingerprints in real-time in the production environment. Moreover, we develop a differential detection model based on device fingerprints to discover stealthy insider attacks efficiently. We evaluate the proposed method in a real-world testbed. Experiment results show that the detecting performance of classifiers approaches 100% when the the number of component types is small.

CCS Concepts: • **Security and privacy → Intrusion detection systems**.

Additional Key Words and Phrases: Device fingerprinting, insider attacks, multi-physical features, ICPS.

1 INTRODUCTION

The proliferation of intelligent computing, sensor networks, and data-driven manufacturing has promoted the development of Industrial Cyber-Physical Systems (ICPS). ICPS-related applications are expected to reach 81 billion US dollars by 2023, with an annual growth rate of 4.9% [12]. In recent years, ICPS have exposed many vulnerabilities [15, 28]. However, ICPS software and hardware suffer from a long updating period. Furthermore, many devices do not have access to critical patches from the manufacturers, leading to a series of security threats [29]. We categorize attacks into *outsider* attacks and *insider* attacks [8, 25]. When adversaries are insiders with certain authorized access to the ICPS, they can perform uncertified acts to launch insider attacks. Highly skilled

Authors' addresses: Zhen Hong, zhong1983@zjut.edu.cn, Zhejiang University of Technology, China; Lingling Lu, lulingling@email.cufe.edu.cn, (corresponding author), Zhejiang University, China; Dehua Zheng, Zhejiang Sci-Tech University, China, zhengdehua6666@gmail.com; Jiahui Suo, Zhejiang University of Technology, China, s15168300159@163.com; Peng Sun, Zhejiang University of Technology, China, sp12138sp@gmail.com; Raheem Beyah, Georgia Institute of Technology, USA, rbeyah@ece.gatech.edu; Zhenyu Wen, Zhejiang University of Technology, China, zhenyuwen@zjut.edu.cn.

1550-4859/2023/8-ART1 $15.00
https://doi.org/10.1145/3582691

adversaries such as terrorists and organized criminals are outsiders who launch attacks from outside ICPS for economic or political purposes [1, 5].

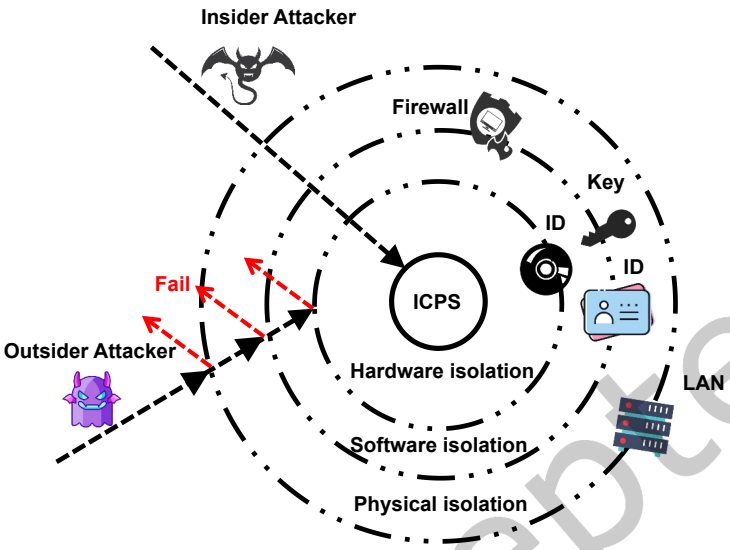


Fig. 1. The existing ICPS protection system

As shown in Fig. 1, the existing ICPS protection system employs **three isolation layers** to prevent security threats. To achieve physical isolation, ICPS disconnect from the public network, establishing an independent and private local network. In the software isolation layer, the software protection fails to recognize an adversary with a legal identity or login key. Hardware isolation prevents the invasion of viruses or Trojans mainly by blocking hardware interfaces such as USB. However, as long as input and output methods exist, like using CD-ROMs for reading and writing, there is a certain probability of infection.

Unfortunately, the above external defenses are not omnipotent. They fail to stop insider attacks launched by staff or business partners who can bypass the protection mechanism, easily accessing the system, network and data with a legal identity [13]. For example, *Stuxnet* [17] was the first worm to compromise Iran's ICPS through an insider attack in 2010, resulting in the scrapping of 20% of Iran's centrifuges and the infection of 30,000 terminals [23]. Furthermore, insiders installed malware that caused a large-scale blackout in the Ukrainian power sector in 2015, bringing significant property losses [24]. An insider attack is a more harmful and insidious behavior than an outsider attack, negatively affecting the confidentiality, integrity, and availability of the ICPS [32].

To address these limitations, we propose a multi-physical features based device fingerprint to enhance current intrusion detection methods in the ICPS environment. On the one hand, fingerprints typically collect hidden equipment features to enable **unique device identification**, allowing fast, accurate detection of insidious attack behaviors [33]. We can fuse some physical device attributes into fingerprints that attackers cannot easily modify. Even if an insider bypasses most external defenses, modifying physical devices' unique identities is impossible. On the other hand, the device fingerprinting technology has **generality** in the intrusion detection of ICPS *control flow*. To discover the attack behavior against a device, we first construct a physical model of the key devices on the *control flow*. We then obtain a set of interlocked physical variables and the critical variable that implements the ICPS function. Combining these physical features as a device fingerprint, we monitor the device fingerprints

for changes in the production environment to discover stealthy insider attacks. In this paper, we use the discrete manufacturing sorting system as an example to illustrate our proposed fingerprinting approach for ICPS intrusion detection.

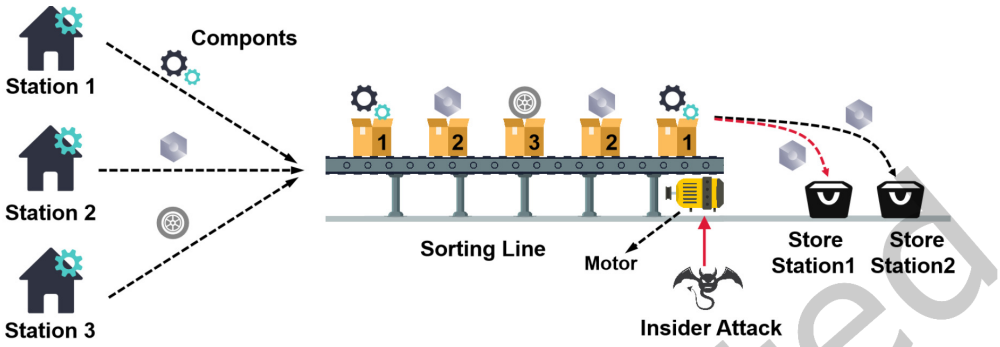


Fig. 2. Crash a sorting system via insider attacks and detect the attacking behavior via fingerprints

As shown in Fig. 2, a sorting line receives different types of components from multiple stations. Then components are sorted into distinct store stations. When the system suffers from an insider attack, the component that should have been sorted to store station 2 is incorrectly sorted to store station 1, resulting in the wrong component classification. If the attacking behavior is undetected, a prolonged attack may lead to a collapse of the whole system, thus affecting industrial production or indirectly damaging hardware devices. Consequently, to detect insider attacks in the sorting system, we propose **multi-physical features based fingerprinting**. However, there remain two challenges to implementing fingerprint-based intrusion detection.

Q1: How to identify key physical device features that construct a fingerprint?
Q2: How to detect insider attacks using fingerprints?

To answer Q1, we establish a physical model of the sorting line and formally analyze key features that influence fingerprinting. In the sorting system, device configurations include the sorting line's height, speed, and other physical variables. Note that fingerprints generated under different device physical configurations are highly separable. If an attacker does not know the exact system device configuration, the fingerprint obtained by forging the sensor data must not match the current device configuration. Therefore, we use a classifier to generate fingerprints and compare them with fingerprints received by sensors to implement a differential detection model. If they are not identical, we can infer that an insider attack has happened and the attacker tampered with the sensor data. Particularly, to solve Q2, we first collect and optimize featurized data from a physical testbed, training a classifier model for fingerprint generation. We then develop a differential detection model employing the fingerprint classifier. The differential detection strategy can tell whether a fingerprint is generated by normal device operations or an insider attacker, providing a solid supplement to the existing protection mechanism in the ICPS environment.

The contributions of our work are summarized as follows.

(1) We propose multi-physical features based fingerprinting to detect insider attacks in ICPS. We formalize the physical model of a sorting system to analyze the critical device features (§4).
(2) We present the workflow of fingerprint synthesis and develop a differential detection model based on device fingerprints to discover insider attacks efficiently. We further realize the feature data optimization methods to train the classifier for fingerprint generation (§5).

(3) We design the r metric to quantify the detection success rate of the detection model. We then solve the detection analysis as a variational problem using functional analysis, demonstrating detection success probability is directly related to the number of device configurations determined by feature combination (§6).
(4) We implement a real physical testbed to simulate the sorting system. Experiments demonstrate the effectiveness of the fingerprint-based differential detection method and the robustness of fingerprint generation (§7).

The rest of the paper is organized as follows. In section 2, we give a brief introduction to existing ICPS fingerprint-based intrusion detection methods. Then, we describe the system model and threat model in section 3, followed by the formalized physical model of the sorting system in section 4. In section 5, we give the details of multi-physical features based fingerprinting methods and propose feature data optimization methods. In section 6, we formally analyze the detection methods. In section 7, we conduct extensive experiments to evaluate the detecting performance of the fingerprinting approach. Finally, we conclude our paper in section 8.

2 RELATED WORK

Some ICPS infrequently update outdated software and hardware, causing significant security risks and vulnerabilities [7]. Many works [9, 16, 18] propose intrusion detection systems (IDS) to monitor the network traffic, finding anomalous data packets. Nevertheless, they ignore the unique characteristics and critical information of ICPS's physical status. Thus, they fail to discover an insider attacker who has already obtained certain *access to the network*.

Recently, some works [6, 10, 22] have leveraged the **physical properties** of ICPS to optimize the intrusion detection scheme. For example, Urbina et al. [30] study physics-based attack detection, which can discover attackers intending to hide ICPS manipulation behaviors. They demonstrate that an appropriate combination and configuration of the detection scheme can mitigate the impact of such stealthy attacks. McParland et al. [21] exploit a specification-based intrusion detection framework to monitor for physical constraint violations. In the case of a boiler with a heater, the framework monitors the boiler's behavior and warns of a possible attack when the temperature exceeds the normal range.

Researchers find **fingerprints** difficult to tamper with and enable unique equipment identification. Many works thus propose constructing device fingerprints to enable attack detection [19, 26]. Below we first introduce some representative fingerprint-based intrusion detection methods. We then make a detailed comparison and illustrate the advantages of our approach.

2.1 Fingerprint-Based Intrusion Detection Methods

Ahmed et al. [3] employ a hardware fingerprint based on the sensor measurements' noise pattern. This hardware identification method is non-invasive and works passively without affecting the functionality of ICPS. Residual is the difference between the system's predicted output and the actual output measurement. Hong et al. [13] observe that the measurement noise and process noise from the water-level system's sensor generates stable residuals. Therefore, they use system operation's noise residuals to build fingerprints and detect attacks. Unfortunately, the above **sensor noise based fingerprinting** methods do not apply to ICPS applications without *significant noise*.

Formby et al. [9] propose two types of device fingerprinting methods. The first method develops accurate fingerprints by measuring the response time of data processing and using the unique network characteristics of ICPS devices. The second method utilizes the physical operation time as a unique signature for each device type. They claim that the physical fingerprinting method achieved an accuracy of 92% in distinguishing real latching relays from counterfeit ones. The authors also show that a forgery attack on fingerprint recognition technology requires a highly skilled attacker with excellent system equipment knowledge. Gu et al. [12] also propose using

device operation timestamps and configurations to generate device fingerprints. However, if the ICPS *operation time* is not a crucial physical feature, then the **timestamp based fingerprinting** methods described above are infeasible.

2.2 Comparison with Fingerprint-Based Intrusion Detection Methods

As discussed above, the passive fingerprinting technology effectively enhances ICPS security without degrading system performance. This paper proposes multi-physical features based fingerprinting to detect insider attacks in ICPS. The **sensor noise based fingerprints** [3, 13] and the multi-physical features based fingerprints all employ stable physical features in their respective systems to construct device fingerprints. Therefore, even if the attack is stealthy and short-lived, the system can detect the attack behavior in time. The difference is that multi-physical features based fingerprinting is the *first* fingerprint technique that works for the discrete manufacturing industry (sorting system).

Compared with **timestamp based fingerprints** [9, 12], multi-physical features based fingerprints also build a formalized physical model to analyze critical features. The difference is that our approach has generality in the intrusion detection of ICPS *control flow*. To discover the attack behavior against a device, we first construct a physical model of the essential devices on the *control flow*. We then obtain a set of interlocked physical variables and the critical variable that implements the ICPS function. Combining these physical features as a device fingerprint, we develop a differential detection model based on device fingerprints to discover insider attacks efficiently. Also, the fingerprinting approach's accuracy, precision, and recall against the real-life dataset show that the detecting performance of classifiers approaches 100% when the device configuration number is small.

3 SYSTEM MODEL AND THREAT MODEL

In this section, we first introduce the workflow of the sorting system in discrete manufacturing and then present the threat model in the sorting system. Finally, we explain an insider attack pipeline without intrusion-detecting methods.

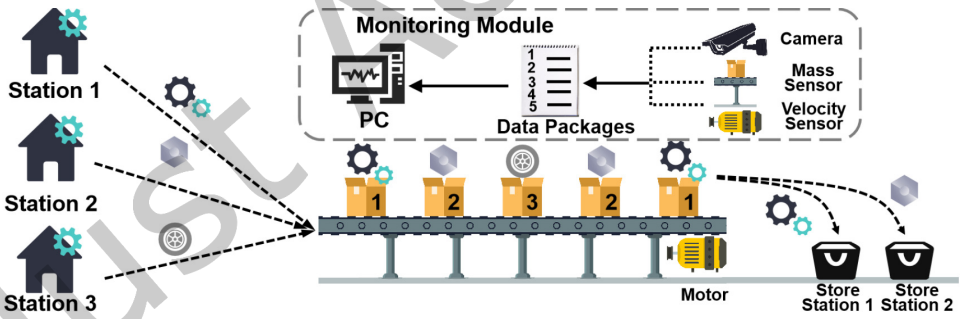


Fig. 3. The workflow of a sorting system

Workflow of the Sorting System. Fig.3 shows a schematic diagram of how a sorting system works. The sorting system aims to categorize different types of components automatically. In Fig.3, each manufacturing station produces one type of component. Different types of components have distinct mass values. Therefore, the components placed on the conveyor belt from different manufacturing stations will be thrown into distinct positions at the end of the conveyor belt. Depending on the position, different types of components can be sorted automatically. Besides, many sensors in the sorting system are responsible for observing and securing the system. Specifically, PC and sensors make up the monitoring module. The *vision sensors* in the camera, the *mass sensors*

in the belt, and *speed sensors* in the motor, etc., will send *data packages* collected in sensor devices to the PC (personal computer) for further analysis.

Threat Model of the Sorting System. Without well-designed intrusion detection methods, the sorting system has a vast attack surface. Therefore we consider two types of attacks: outsider attacks and insider attacks. As discussed in §2 related work, outsider attacks have been extensively studied. This paper focuses on detecting the more stealthy and challenging insider attacks because insiders have detailed information about ICPS workflows, leading to the potential for significant security losses [25]. We consider three insider attack methods against physical, software, and hardware isolations to bypass external defenses and achieve *real-time access* to ICPS devices.

(1) The adversary cannot get physical access to devices but has the authority to access the local internal network, thus launching attacks.
(2) The adversary can utilize vulnerabilities to intrude into software and thus execute attacks.
(3) The adversary can touch various field devices and inject Trojan or virus through hardware interfaces, e.g., USB, to launch attacks.

Furthermore, we illustrate the above three insider attack methods in *realistic* ICPS scenarios. (1) For the first method, when an ICPS employee is intentionally or unintentionally infected with a virus, he launches an attack on a live device via the local internal network like Havex [2]. (2) For the second method, a malicious staff with professional hacking skills can utilize vulnerabilities to bypass software isolation and execute attacks [1]. (3) For the last method, the field devices may be prone to suffering a potential attack because an employee inserts an infected USB, e.g., Stuxnet [17]. Consequently, even if malicious attackers do not have legal access, they exert themselves to the utmost to enter the internal network or directly access the physical system, aiming at launching an insider attack. Therefore, it is difficult to prevent such attacks in the realistic scenario completely.

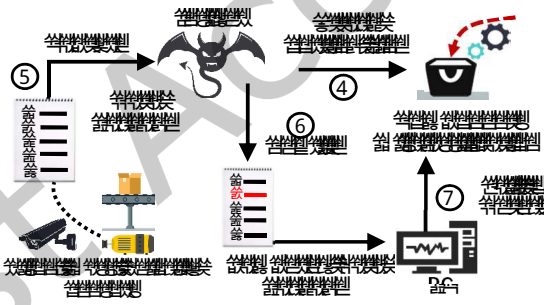


Fig. 4. The workflow of an insider attack in the sorting system

Pipeline of an insider attack. Fig.4 illustrates the pipeline of an insider attack in the sorting system without intrusion-detecting methods. In ①, the insider attacks the sorting line, leading to the misclassification of components (see detailed attacking methods in Section 5.2). In ②, the attacker captures data passed from sensors to the PC, then tampers with the sensor data. Note that vision sensor in the camera collects component displacement, the mass sensor collects component mass, and the speed sensor collects belt velocity provided by the motor. In ③, the attacker rewrites the original data package with the tampered data. In ④, when the PC receives the data package, it *fails to detect* the misclassification of components and such attacking behavior. Because, according to the data obtained by the PC, different types (mass) of components are correctly sorted into their corresponding positions, the PC does not detect any anomalies in the system. The insider attack thus succeeds. Furthermore, the sorting system will collapse in the long run, causing significant damage.

To discover insider attacks, we develop accurate fingerprinting methods (more details in §5) to detect tampered data packets and other attack behaviors, reducing the probability of sorting errors.

4 FORMALIZED PHYSICAL MODEL OF THE SORTING SYSTEM

To detect insider attacks, we formalize the *physical model* of the sorting system and analyze the key features that influence fingerprinting in this section. A device's fingerprint is primarily related to its unique *physical characteristics* and hardware configurations [12]. Besides, in §5 we train a *classifier*, obtaining a more accurate physical model for displacement calculation and fingerprint generation.

4.1 Key Physical Variables for Fingerprinting

To address the malicious sabotage of sensor data in sorting systems, we propose a fingerprinting method by combining four physical variables, i.e., mass, velocity, height, and displacement in the sorting system. We have **two observations** from the formalized physical model of the sorting system. (1) Firstly, these physical variables are closely related to the inherent physical characteristics of each sorting system. (2) Secondly, these four physical variables are interlocked. Precisely, if three of the four variables are known, we can calculate the unique value of the remaining variable via a mathematical equation obtained from the physical model. Based on the above observations, we leverage **four physical variables** to generate fingerprints for the sorting system, providing intrusion detection capabilities against insider attacks. We will elucidate the fingerprinting approach in §5. We formalize the physical model below.

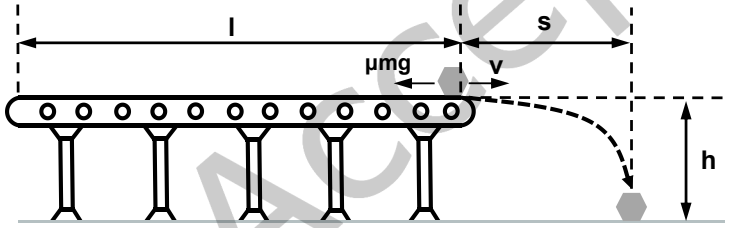


Fig. 5. The physical mode abstracted from the sorting system

Definition 1 (Relationship between physical variables). *We describe the physical model abstracted from the sorting system in Fig.5. We clarify the relationship between the displacement (s) of components and other variables in Equation (1). Specifically, m is the mass of component, v denotes the velocity of moving object, h denotes the height of the conveyor and s is the distance that the component falls onto the table after being thrown from the conveyor belt.*

$$s \sim f(m, v, h) \tag{1}$$

Equation (1) demonstrates that each parameter is sensitive and essential for the physical model of the sorting system. Specifically, the displacement (s) of a component is closely related to mass of a component (m), moving velocity (v), and height of the conveyor (h). Because there is a matching relationship among the four physical variables, the system can easily detect attackers' malicious sabotage of sensor data. Next, we elaborate on the specific relationship of these four physical variables from a **kinematic perspective**.

Definition 2 (Belt friction). *In the motion of the conveyor belt, if the component is stationary relative to the belt, then the belt friction will provide the forwarding force to the component. Here, the belt friction (f_{belt}) is*

$$f_{belt} = \mu mg \tag{2}$$

Definition 3 (Component friction). *In Equation (2), g is the gravitational acceleration, μ denotes the coefficient of belt friction which is determined by the physical characteristics of the belt. According to Theorem 1 which describes the displacement formula of uniformly accelerated motion [11], the* maximum *velocity (v_{max}) that an component can obtain through friction is*

$$v_{max} = \sqrt{2\mu g l} \tag{3}$$

In Equation (3), l is the length from the initial placement point of the component to separation point (the end of the conveyor belt in Fig. 5). For example, l is the length from the position where different manufacturing stations' components first reach the conveyor belt to the end position of the belt where components fall.

There are two **situations** when a component is moving on a conveyor belt in Fig. 5.

- **Situation A**: The component is moving at the same velocity as the conveyor belt (v_{belt}).
- **Situation B**: The velocity of the component (v) does not reach the velocity of the conveyor belt, i.e., the component is moving slower than the conveyor belt.

In Equation (3), $v_{max} = \sqrt{2\mu g l}$, therefore, given the condition $v = v_{belt}$, we have $v_{belt} \leq \sqrt{2\mu g l}$. On the other hand, when given the condition $v < v_{belt}$, we can naturally come to this conclusion: $v_{belt} > \sqrt{2\mu g l}$. In these two situations, forces on the components are different; therefore, the formalized physical models also differ. We carefully investigate both models to find the critical features for fingerprinting.

4.2 Component Displacement Calculation Under Situation A

We first consider the situation where the component is moving at the same velocity as the conveyor belt and $v_{belt} \leq \sqrt{2\mu g l}$. Fig. 5 shows a straightforward physical model for sorting. In an actual sorting process, we cannot ignore the air resistance.

Definition 4 (Air resistance). *Air resistance (f_{af}) is defined in Equation (4), k is the coefficient of air resistance. k is usually a constant related to the characteristic area (windward area), the smoothness and the overall shape of the component.*

$$f_{af} = kv^2 \tag{4}$$

Definition 5 (Component's resultant force). *If we only consider the free-fall motion of the component, the component's resultant force (F) in the direction of vertical falling is*

$$F = mg - f_{af} = mg - kv^2 \tag{5}$$

Definition 6 (Component's acceleration). *In the free-fall motion, when a component falls freely, its acceleration (a) is*

$$a = \frac{F}{m} = g - \frac{kv^2}{m} = \frac{\mathrm{d}v}{\mathrm{d}t} \tag{6}$$

The acceleration (a) in Equation (6) is also the amount of change in velocity (Δv) per unit time (Δt), i.e., $a = \frac{\Delta v}{\Delta t} = \frac{\mathrm{d}v}{\mathrm{d}t}$. We can observe from Equation (4) and Equation (6) that when the component's velocity (v) in the direction of vertical falling increases, the component's acceleration (a) decreases and air resistance (f_{af}) increases. Consequently, the component's acceleration is not a constant. However, in the discrete manufacturing scenario, the height of the component's fall (h) and the component's velocity in the direction of vertical falling are small, the falling time (t_{all}) is short. Therefore, we **approximate** the air resistance f_{af} as a **constant** f_c, simplifying the component's displacement calculation.

In this case, the resultant force in the direction of vertical falling is $F = mg - f_c$ and the acceleration of the component is $a = g - \frac{f_c}{m}$. Utilizing Theorem 1, we can obtain the total time (t_{all}) taken for a component to fall from the conveyor belt to the ground in Equation (7).

$$t_{all} = \sqrt{\frac{2h}{g - \frac{f_c}{m}}} \tag{7}$$

Consequently, the displacement (s) that the component moves forward during the t_{all} can be calcuated in Equation (8).

$$s = \sqrt{\frac{2hv_{belt}^2}{g - \frac{f_c}{m}}} - \frac{hf_c}{mg - f_c} \tag{8}$$

Theorem 1 (Displacement formula of uniformly accelerated motion). *When the acceleration a of an object is constant, Equation (9) illustrates the relationship between the displacement s of the object's motion, the initial object velocity v, the acceleration a, and time t [11].*

$$s = vt + \frac{1}{2}at^2 \tag{9}$$

Note that the proof of Equation (3), Equation (7) and Equation (13) use Theorem 1.

4.3 Component Displacement Calculation Under Situation B

We then consider the situation where $v < v_{belt}$ and $v = \sqrt{2\mu gl}$. As shown in Fig. 6, the component will undergo an atypical horizontal parabolic motion when the belt speed is low and the belt friction is significant. To better visualize θ in Fig. 6, we perform a force analysis on the object at the critical point of falling from the belt. The belt friction force at this point (μmg) is less than the component force of gravity in the object's falling direction ($mg\cos\theta$), satisfying Equation (10). Therefore, we can calculate the angle θ between the direction of an object falling and the vertical line based on the belt friction coefficient μ. For example, when $\mu = 0.1$, we can get $\cos\theta > 0.1$ and thus calculate $\theta < 84°$ via Equation (10). In the subsequent displacement calculation, we still use θ instead of a specific angle in the equations.

$$\mu mg < mg\cos\theta \tag{10}$$

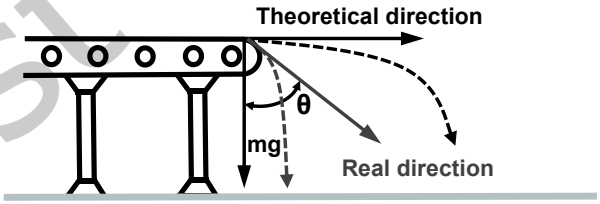


Fig. 6. The atypically horizontal parabolic motion

To calculate component's acceleration (a') along the vertical direction in Fig. 6, we first calculate the resultant force (F') of the component in Equation (11). Specifically, $\mu mg\cos\theta$ is the resolve force of the belt friction force along the vertical direction and f_c is the air resistance. Thus, component's acceleration (a') is given in Equation (12).

$$F' = \mu mg\cos\theta + mg - f_c \tag{11}$$

$$a' = \mu g\cos\theta + g - \frac{f_c}{m} \tag{12}$$

Utilizing Theorem 1 which describes the displacement formula of uniformly accelerated motion, we can obtain the total time (t'_{all}) taken for a component to fall from the conveyor belt to the ground in Equation (13).

$$t'_{all} = \sqrt{\frac{2h}{\mu g \cos\theta + g - \frac{f_c}{m}}} \tag{13}$$

Subsequently, we obtain the displacement (s') that the component moves forward during t'_{all} in Equation (14).

$$s' = \sqrt{\frac{4\mu g l h}{\mu g \cos\theta + g - \frac{f_c}{m}}} - \frac{h f_c}{\mu g \cos\theta + g - \frac{f_c}{m}} \tag{14}$$

4.4 The Displacement Calculation Formula

We merge Equation (8) and (14) to get a complete displacement calculation formula in Equation (15). We find that the final displacement is related to the parameters such as v_{belt}, h, m, μ and f_c from Equation (15). We consider the belt friction coefficient μ and air resistance f_c constants. Therefore, we leverage four physical features (mass, velocity, height, and displacement) to generate unique fingerprints for the sorting system.

$$s = \begin{cases} \sqrt{\frac{2hv_{belt}^2}{g - \frac{f_c}{m}}} - \frac{hf_c}{mg - f_c} & v_{belt} \le \sqrt{2\mu g l} \\ \sqrt{\frac{4\mu g l h}{\mu g \cos\theta + g - \frac{f_c}{m}}} - \frac{hf_c}{\mu g \cos\theta + g - \frac{f_c}{m}} & v_{belt} > \sqrt{2\mu g l} \end{cases} \tag{15}$$

5 MULTI-PHYSICAL FEATURES BASED FINGERPRINTING METHODS

The previous §4 proves that displacement, mass, velocity, and height are interlocked physical variables. Additionally, these variables represent the unique physical characteristics of ICPS devices and can be used to generate physical fingerprints. In this section, we first illustrate the multi-physical features based fingerprinting methods. We then describe the insider attack in the sorting system, clarifying the capabilities of insiders and explaining how to detect attacking behaviors using the differential detection strategy. Finally, we propose an optimization method for physical feature data to train the classifier for displacement calculation and fingerprint generation.

5.1 Fingerprint Generation Overview

In this paper, we propose to utilize multiple physical features of ICPS devices in the physical layer for fingerprint generation. Specifically, we employ the sorting system as a typical ICPS example to clarify the overall workflow of fingerprint generation. Fig. 7 shows the workflow of fingerprint synthesis. The square represents mass data, the triangle represents velocity data, the diamond represents height data, and the round represents displacement data. It's worth noting that different colors represent different data. These characteristics data represent the features of the sorting system in the physical layer. We present the detailed workflow below.

① Various sensors collect raw data of features from the sorting system.

② PC preprocesses the collected data. We can gain multiple physical features from the preprocessed data, e.g., mass data (square), velocity data (triangle), height data (diamond) and displacement (round). In the case of the sorting system, each component corresponds to a set of features obtained by sensors. PC classifies multiple physical features into corresponding pairs, e.g., (m, v, h, s). We utilize the minimum deviation based and confidence threshold based optimization methods to eliminate interfering noise in feature data (see §5.3 for details). We then employ optimized features as the training dataset for the classifier model.

③ In the **model training pipeline**, we use many pre-collected feature pairs like (m_1, v_1, h_1), (m_2, v_2, h_2) as the input to a physical model and corresponding displacement s_1, s_2 as the output of the physical model,

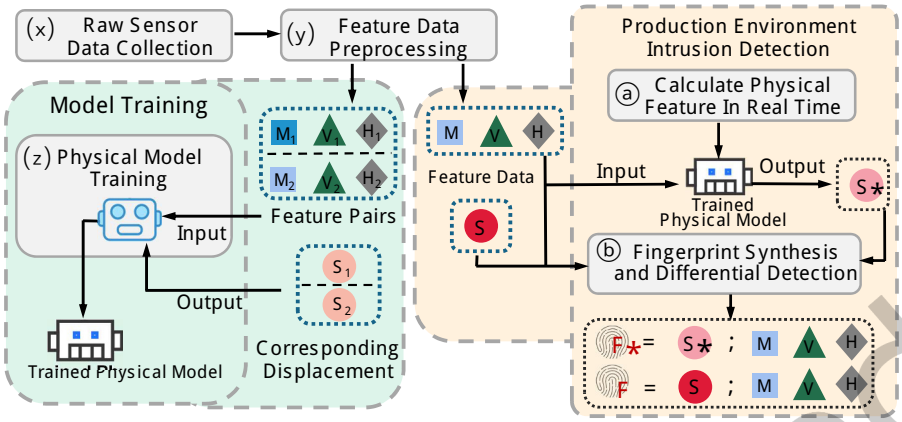


Fig. 7. Workflow of fingerprint generation with multi-physical features

training a classifier to get a more accurate physical model. Note that the trained classifier will be used in the differential detection pipeline.

④ In the **production environment**, we use the feature pairs (m, v, h) as the input to the classifier (trained physical model) and calculate the output displacement s_* in real time.

⑤ In the fingerprint synthesis and **differential detection** step, the differential detection strategy can tell whether a fingerprint is generated by normal device operations or an insider attacker. Specifically, we first synthesize multiple physical features (m, v, h, s) obtained from **sensor data** to generate fingerprint F of the current ICPS devices. We then combine the classifier output s_*, utilizing (m, v, h, s_*) to form a fingerprint F_*. If $F \neq F_*$, we detect attacking behavior in the system. Because the attacker may have tampered with the **sensor data**, resulting in $F \neq F_*$.

The underlying reason for applying fingerprint recognition in intrusion detection is that fingerprints generated under different device physical configurations are highly separable. If an attacker does not know the exact system device configuration, the fingerprint obtained by forging the sensor data must not match the current device configuration (more empirical details in §7.2). Next, we specify how to train the classifier model in Fig. 7.

Physical model training with real data inputs. We can calculate the component displacement with corresponding velocity, mass, and height data with the help of Equation (15). However, the physical model has various simplifications and estimations. Therefore, in order to obtain a *more accurate* physical model to calculate component displacement, we propose using real-life data collected from ICPS sensors to train a *classifier*. The input X and output Y of the classifier model and the training set T are formulated as follows,

$$\begin{cases} X = \left\{ x_j | \left({x_j}^{(1)}, {x_j}^{(2)}, {x_j}^{(3)} \right)^{\mathrm{T}} \right\} \\ Y = s_j, j \in (1, 2, 3, \ldots) \\ T = \{(x_i, y_i) | (x_1, y_1), (x_2, y_2), \ldots, i \in (1, 2, 3, \ldots)\} \end{cases} \tag{16}$$

In Equation (16) x_j denotes the j-th physical feature pairs of the input, ${x_j}^{(1)}, {x_j}^{(2)}, {x_j}^{(3)}$ represents the three specific physical features contained in the j-th feature pairs. For example, the feature pairs x_j are $(m_1, v_1, h_1), (m_2, v_2, h_2)$ in the sorting system, and ${x_j}^{(1)}$ is m_1. s_j is the corresponding classifier model output for each feature pair of inputs, such as displacement s_1 and s_2. (x_i, y_i) in the training set T denotes a sample point which represents the physical characteristics of a component and its corresponding real displacement.

We obtain the training data from velocity, mass, height, and vision sensors in physical testbed experiments. We take mass, velocity, and height as inputs of the classifier. We also feed the matching displacements as outputs to the classifier. Note that the selection of **classifier** is user-customized, referring to **predictive models** such as decision trees [27]. Distinct ICPS can choose different classifier models for training. The output displacements of the classifier are then fitted to the real displacements collected from sensors, aiming at training a physical model with high classification accuracy. After the classifier reaches the accuracy threshold required by the ICPS, we use the trained classifier to generate displacements in the production environment in real time. The multi-physical features (classifier input) and newly gained classifier output (i.e., displacement) are synthesized to generate a fingerprint of the current device.

5.2 Insider Attacks and Fingerprint-Based Differential Detection Methods

We describe how to detect insider attacks with multi-physical features based fingerprints in this section.

The detailed insider attack workflow. In §1, Fig. 2 gives an overview of insider attacks in the sorting system. Specifically, different types of components' masses are distinct. Therefore, the components will be thrown into different positions at the end of the conveyor belt under the same device configuration (belt velocity and height). Depending on the position, different types of components are sorted automatically. The adversary **aims** to disrupt the sorting lines without **being detected** by the system. We specify two critical steps for the insider attack against the sorting system.

① When a component moves to a position near the end of the belt, the insider controls the motor, changing the v_{belt} for a short period, i.e., from v_{origin} to v_{new}. When v_{belt} is v_{origin}, the component displacement is s_{origin}. When v_{belt} is v_{new}, the component displacement is s_{new}. The component thus will be thrown into a different position, causing the component to be misclassified. The velocity will then quickly return to v_{origin} without being noticed by sensors. The attacker also tampers with mass sensor data, changing m_{origin} to m_{new} that satisfies the new component classification. Specifically, in Fig. 2 after an insider attack, the component that should have been sorted to store station 2 is sorted to store station 1.

② Moreover, ICPS also cannot detect the attack or identify incorrectly sorted components without fingerprints. Because to avoid being detected by the system, the insider attacker (1) captures data packages from velocity and mass sensors and (2) rewrites the original data package with the tampered data, causing the PC to receive incorrect data.

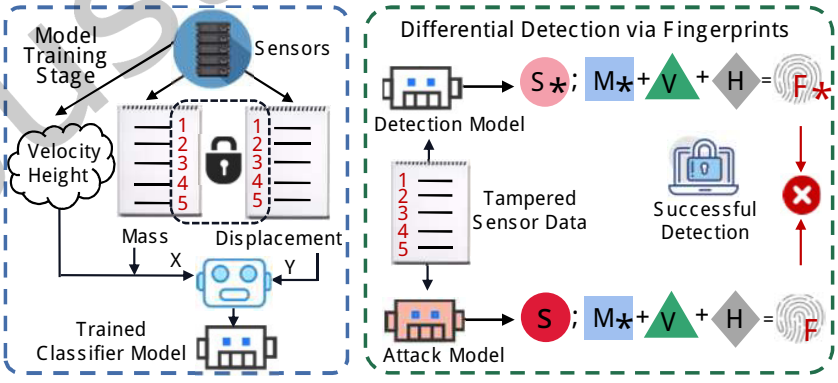


Fig. 8. Fingerprint-based differential detection method against the insider attack.

Capabilities of an insider. Firstly, we clarify that the fingerprint-based intrusion detection method is **valid under** the limited attacker capability definition. Specifically, attacker is not a strong attacker who can intercept all the packets in the system. The data packages intercepted by the insider attacker are black-box data, and there may be a *partial loss* of sensor data. Therefore, the attacker does not know the targeted attacking device's *exact configuration* or the classifier model's full parameters. Notably, the above capability definition has **practical significance**. In many scenarios, the attacker does not have a strong capability but still launches harmful and insidious attacks. For example, in *Stuxnet*, the attacker cannot access all packets in the ICPS but has compromised a vulnerable device in the network with malware [13].

Secondly, we illustrate how an insider trains an **attack model** to forge fingerprints in Fig. 8. Note that, in the case of sorting system, one sorting line has a fixed **device configuration**, i.e, the belt height and velocity. In the model training stage, the detection system uses many pre-collected optimized feature pairs like (m_1, v, h), (m_2, v, h) as inputs to a classifier and corresponding displacement s_1, s_2 as outputs of the classifier, training the model to get a more accurate **detection model** (trained classifier). However, the insider trains the **attack model** under two conditions. ① Insider *randomly guesses* the classifier model parameters (belt height and velocity) from limited intercepted sensor data. ② Insider cannot know precisely the belt speed v_{origin} and v_{new} defined in the above attack workflow.

Differential detection via fingerprints. The key idea of the differential detection strategy relies on passing the same input to the classifier and comparing the output [20, 31]. In the example of Fig. 8, the attacker tamper with sensor data to cover up the attacking behavior. The **detection model** (trained classifier model) output s_*, generating F_* with input (m_*, v, h). The attacker forges m_* in the sensor data that can output s via the **attack model**, generating F with input (m_*, v, h). Applying the differential detection strategy, if F_* equals F, the system assumes no attack event has occurred. Otherwise, we can detect intrusion via fingerprints.

The key, therefore, lies in the attacker correctly guessing the *configuration of the physical device*. Besides, feature combination determines the number of device configurations. In the example of Fig. 8, if there are four velocity configurations and two height configurations, then the attacker has only $\frac{1}{8}$ ($\frac{1}{4\times2}$) probability of successfully guessing the full parameters of the classifier model and launching an attack on the system. Heuristically, we can conclude that the greater the physical device configurations number, the smaller the probability that the attacker successfully performs an attack.

5.3 Optimization Methods for Feature Data

We can utilize the optimization method in the physical model training stage (see §5.1 for training details) to eliminate interfering noise in feature data and train a physical model with high classification accuracy.

When there is a large amount of sensor data, some abnormal data will inevitably exist. Consequently, we propose to optimize data for interfering noise elimination, leading to a better visualization of the feature data distribution. We can then use the optimal data filtered from a large batch of data as fit points for the visualized data distribution graph. For example, in Fig. 13, the component displacements of different masses at the same belt speed (device configuration) are fitted by a curve.

Our goal is to minimize the Euclidean distance between the displacements generated by components under the same physical conditions (e.g., the same mass, velocity, and height) in the example of the sorting system. Equation (17) and (18) illustrate the deviation and threshold metrics of the optimization methods.

$$S_{dev}(s_{op}) = \sqrt{\frac{\sum_{j=1}^{N} (s_{op} - s_j)^2}{N-1}} \qquad , op \neq j \tag{17}$$

The displacement optimization method based on minimum deviation. In Equation (17), s_{op} denotes the target solution for optimal displacement under a particular physical condition (v, h, m). s_j denotes multiple displacement data generated under the same physical conditions as s_{op}. We conduct the experiments N times and obtain multiple displacement data that are collected under the same physical conditions. $S_{dev}(s_{op})$ denotes the standard cross deviation function for calculating displacements difference among collected data. In order to solve for the optimal target displacement s_{op} value, we need to iterate through all the s_j. The s_{op} that minimizes $S_{dev}()$ will be the optimal displacement under a particular physical condition. The optimization method calculates the optimal displacement under a specific physical characteristics configuration, making the feature data more accurate and facilitating the data fitting in the experiment 7.2.

Algorithm 1 The optimization method for displacement features

Input: A set of displacement data $D_{displacement}$
Output: Steadily optimal displacement set S_{smooth}

$S_{dev} = [\,]$ ▷ The displacement optimization method based on minimum deviation
$S_{op} \leftarrow \emptyset$
for $S_{displacement}$ **in** $D_{displacement}$ **do**
  for s_i **in** $S_{displacement}$ **do**
    $s_{dev} \leftarrow 0$
    for s_j **in** $S_{displacement}$ **do**
      $s_{dev} \leftarrow s_{dev} + (s_i - s_j)^2$
    end for
    $s_{dev} \leftarrow \sqrt{\frac{s_{dev}}{(len(S_{displacement})-1)}}$
    $S_{dev}(s_i) \leftarrow s_{dev}$
  end for
  $s_i \leftarrow \min S_{dev}(s_i)$
  $S_{op} \leftarrow S_{op} \cup s_i$
end for ▷ The displacement smoothing method based on confidence threshold
$S_{op}.sort(), S_{op} = s_{op-1}, s_{op}, s_{op+1}, \ldots$
$S_{smooth} \leftarrow \emptyset$
for s_{op} **in** S_{op} **do**
  if $\frac{|s_{op+1}-s_{op}|}{s_{op}} < \sigma$ **then**
    $S_{smooth} \leftarrow S_{smooth} \cup s_{op}$
  end if
end for
return S_{smooth}

$$\sigma_s = \frac{\left|s_{op+1} - s_{op}\right|}{s_{op}} \tag{18}$$

The displacement smoothing method based on confidence threshold. We also propose a confidence level-based displacement smoothing method to accurately obtain the stable displacement from displacement samples. Specifically, we first *sort* the optimal displacement s_{op} of all samples. Then we calculate the displacement deviation σ_s of adjacent samples, and judge whether it exceeds the pre-defined confidence threshold σ. In Equation (18), s_{op+1}

and s_{op} are the two adjacent optimal displacements. For example, if the deviation σ_s is less than 5% (confidence threshold σ), i.e., $\sigma_s < \sigma = 5\%$, then the optimal displacements from the fitting points are in steady value.

Algorithm 1 shows the displacement optimization procedure. The algorithm inputs is a list of displacement data $D_{displacement}$ collected under different device configurations. Each element in $D_{displacement}$ is $S_{displacement}$. $S_{displacement}$ contains multiple displacement data s_i that are collected under the same physical conditions. The algorithm output is a set of steadily optimal displacement data S_{smooth}. We can divide the algorithm into seven steps.

① Initialize a standard cross deviation list S_{dev}, a standard cross deviation element s_{dev} that will latter be added into S_{dev}, and a optimal displacement data set S_{op} (line 1, 2 and 5).
② Traverse the $S_{displacement}$ of displacement data under the same physical condition and calculate the Euclidean distance s_{dev} of s_i from the other s_j (line 4-8).
③ Calculate s_{dev} according to Equation (17) and add it to S_{dev} (line 9-10).
④ The displacement data s_i that minimize the deviation function $S_{dev}()$ will be the optimal displacement s_{op} under a particular physical condition. Append s_i to the optimal displacement data set S_{op} (line 12-13).
⑤ After sorting S_{op}, we have $S_{op} = s_{op-1}, s_{op}, s_{op+1}, \ldots$ (line 15).
⑥ Iterate through the optimal displacement distances and calculate the deviation between adjacent optimal displacement distances according to Equation (18). Determine whether the deviation $\frac{|s_{op+1}-s_{op}|}{s_{op}}$ is smaller than the pre-defined confidence threshold σ. If the deviation is smaller, append the optimal distance s_{op} to the set S_{smooth} (line 17-21).
⑦ Return the set of smooth displacements S_{smooth} (line 22).

5.4 The Generality of Multi-Physical Features Based Fingerprinting

We mentioned in §1 that the device fingerprinting technology has **generality** in the intrusion detection of ICPS *control flow*. By constructing a physical model of the key devices on the *control flow*, we obtain a set of interlocked physical variables and the critical variable that implements the ICPS function. Combining these physical features as a device fingerprint, we monitor the device fingerprints for changes in the production environment to discover stealthy insider attacks.

We take the following two ICPS applications as examples, illustrating how to use the proposed fingerprint technology in other scenarios.

- In the example of electric mixers, the mixing fluid viscosity, mixer depth, motor horsepower, and mixer fan blade's angular velocity are interrelated physical features. The mixer fan blade's angular velocity is the **critical** physical variable to implement the ICPS function (agitation of the liquid). Therefore, in this application, we can use these four physical features of electric mixers as the mixing device's fingerprint.
- In the example of temperature-sensitive latching relays, magnetic ring temperature, the number of turns in the solenoid, current, voltage, and operation response time are interrelated physical features. The latching relay's operation response time is the **critical** physical variable to represent the ICPS function (open or close operation). Therefore, in this application, we can use these five physical features of latching relays as its device fingerprint.

6 INTRUSION DETECTION METHODS ANALYSIS

In this section, we first propose the r metric to quantify the detection model's **detection success rate** against the insider attack that randomly guesses physical parameters. We then solve the detection analysis as a variational problem using functional analysis, demonstrating the influence of device configuration number and probability distribution on the **detection success probability**.

6.1 The r Metric for Quantifying Detection Success Rate

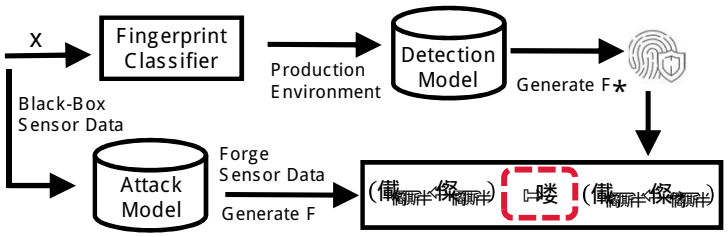


Fig. 9. Use a detection model to discover attacking behavior

Fig. 9 shows the how to use a detection model to discover insider attacks. The **fingerprint classifier** is an accurate physical model trained with real-life data X collected from ICPS sensors. We split the real-life dataset into training data and testing data. The fingerprint training and testing data are generated independently of the joint probability distribution $P(Y, X)$. After we learn the classifier from the real-life dataset, we can use it as the **detection model** in the production environment and define it as a decision function $Y{=}f(X)$. The following gives the calculation process of the r metric.

- An adversary builds an **attack model** by intercepting black-box sensor data, training the classifier model with **randomly guessed** physical parameters. The attacker intercepts the input x_{N+1} by the black box and then gains an output y'_{N+1} through the attack model. The attacker construct the input and output to forge a fingerprint F aiming at cheating the detection model.
- As for the **detection model**, when the input is x_{N+1}, the corresponding output is y_{N+1}, generating F_* for the expected component. If (x_{N+1}, y_{N+1}) is not equal to (x_{N+1}, y'_{N+1}), the system **succeeds in detecting** the attacking behaviors.

$$L = \begin{cases} 0, & (x_{N+1}, y_{N+1}) = (x_{N+1}, y'_{N+1}) \\ 1, & (x_{N+1}, y_{N+1}) \neq (x_{N+1}, y'_{N+1}) \end{cases} \tag{19}$$

$$r = \frac{1}{N} \sum_{i=1}^{N} L \tag{20}$$

In order to quantify the detection model's detection success rate against the attack model that randomly guesses physical parameters, we use a metric function in Equation (20). L in Equation (19) represents the number of successful detections in N attacks. Obviously, the higher the r metric's value, the better the performance of a detection model.

6.2 Analysis of Detection Success Probability

We analyze the device configuration probability distribution's effectiveness on the detection model performance, demonstrating that the larger the number of device configurations, the more effective the detection.

Probability distribution of device configurations. As described in §5.2, the attacker does not know the exact physical model parameters (device configurations). Thus the attacker can only guess the model parameters randomly based on the black box data and partially lost data intercepted from the ICPS. The multi-physical features based fingerprint intrusion detection method only fails when the attacker correctly guesses the physical parameters of the target device. The probability distribution may vary for each device configuration in the ICPS, leading to two critical questions, Q1 and Q2. For example, there are two sorting lines with configuration velocity

v_1 and height h_1. One sorting line is configured with velocity v_2 and height h_1. Then the probability of the configuration (v_1, h_1) occurring is $\frac{2}{3}$.

Q1: How does the probability distribution of device configurations affect the success probability of an insider attack?

Q2: What probability distribution minimizes the success probability of an insider attack?

To answer Q1, we give a formal Definition 8 to measure the impact of device configuration's probability distribution on the insider attack success probability. To answer Q2, We prove Theorem 2 by employing functional analysis. Proof results show that when the probability of N_{config} device configurations' occurrence is uniform, the insider attack has the smallest success probability.

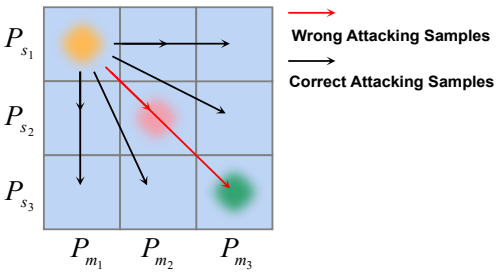


Fig. 10. In the case of discrete sample points, attacking sample point selection determines whether the insider attack can succeed.

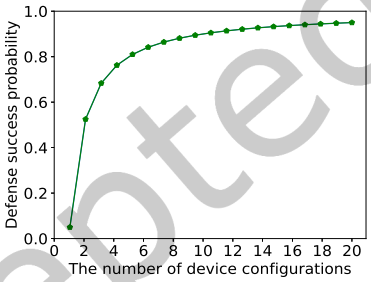


Fig. 11. Detection success probability

Probability distribution of sample points. In the example of the sorting system, the component displacement data is s_* after the attack misclassifies the component. The insider needs to forge a component mess m_* that matches sensor data s_*. Besides, there is some fluctuation and bias in the data collected by the sensors in the system. Furthermore, the overall data distribution matches a Gaussian distribution. The yellow point in Fig. 10 represents the sample points (s_1, m_1). The darker color in the middle of the sample point indicates most of the sample data. The lighter color at the edges of the yellow point shows a small proportion of sample data with fluctuations.

Assuming the attacker wants to tamper with the original correct fingerprint (s_1, m_1) without being detected, the attacker only has two choices, red point (s_2, m_2) or green point (s_3, m_3) in Fig. 10. Selecting another sample point such as (s_1, m_2) or (s_1, m_3) rather than (s_2, m_2) and (s_3, m_3) will lead to failed detectable attacks. Additionally, the red arrow in the nine-box grid represents the selection of correct samples, and the black arrows represent the selection of wrong attacking samples.

How do attackers choose samples? When the attacker makes a guess for the device configuration (velocity v and height h), the attacker will have an attack model that predicts the relationship between m and s (see §5.2 for details). The attacker then forges a matching s_* to $m*$ based on the intercepted target component data, thus selecting sample points (s_n, m_n) such as (s_1, m_2) and (s_2, m_2) in Fig. 10.

Note that, if the probability distribution of device configurations (velocity v and height h) is not uniform, then the probability distribution of the sample points (s, m) is also not uniform. A displacement value s necessarily corresponds to a mass value m. Therefore the probability distribution functions of displacement and mass are the same. Assuming the probability distribution function of displacement is P. Probability of selecting s_1 is p_{s_1}. Probability of selecting m_2 is p_{m_2}.

Definition 7. *The attack success probability is:*

$$P_{attack} = \sum_{i=1}^{n} p_i \sum_{i \neq j}^{n} {p_j}^2 \quad s.t. \quad \sum_{i=1}^{n} p_i = 1 \tag{21}$$

Proof. We prove P_{attack} in Definition 7. Assuming the target attacking sample is (s_1, m_1), a successful attack requires selecting (s_2, m_2) or (s_3, m_3) sample point. In this way, Equation (22) gives the attack success probability $p_{attack}(s_1, m_1)$. Similarly, when the target attacking sample is (s_2, m_2), Equation (23) gives the attack success probability $p_{attack}(s_2, m_2)$. Therefore, we can use Equation (21) to calculate the overall attack success probability p_{attack}. The constraint of Equation (21): assuming the probability of selecting the i-th sample point is p_i, then the probability sum of all sample points is 1.

$$p_{attack}(s_1, m_1) = p_{s_1} \times (p_{s_2} \times p_{m_2} + p_{s_3} \times p_{m_3}) \tag{22}$$

$$p_{attack}(s_2, m_2) = p_{s_2} \times (p_{s_1} \times p_{m_1} + p_{s_3} \times p_{m_3}) \tag{23}$$

□

Definition 8. *When the probability distribution of device configuration is $P(x)$, the attack success probability is:*

$$P_{attack} = \int_x P(x) \int_z P(z)^2 \mathrm{d}z\mathrm{d}x \quad s.t. \quad \int_x P(x) = 1 \int_z P(z) = 1 \tag{24}$$

To answer Q_1 and simplify the P_{attack} calculation in Equation (21), we convert P_{attack}'s summation process to a integration process based on the probability density function. Therefore, we transform Equation (21) to Equation (24). In Equation (21), p_i is a probability, and x_i is the corresponding discrete random variable, i.e., the event of selecting the i-th sample point. In Equation (24), $P(x)$ is a probability density function, and x is the corresponding continuous random variable. In the practical scenario of the sorting system, $P(x)$ is the probability distribution of device configurations. Moreover, the probability sum of all sample points is 1. Consequently, the constraint is $\int_x P(x) = 1$, i.e., the probability density function's full domain integration is 1.

Definition 9. *The probability density function $P(x)$ that minimizes p_{attack} defined in Equation (24) is:*

$$\arg\min_P \int_x P(x) \int_z P(z)^2 \mathrm{d}z\mathrm{d}x \quad s.t. \quad \int_x P(x) = 1 \int_z P(z) = 1 \tag{25}$$

$$= \arg\min_P \int_z P(z)^2 \int_x P(x) \mathrm{d}x\mathrm{d}z \tag{26}$$

$$= \arg\min_P \int_z P(z)^2 \mathrm{d}z \tag{27}$$

To answer Q_2, we solve for the probability density function $P(x)$ that minimizes p_{attack} in Equation (25). Specifically, $\left(\arg\min_P\right)$ means to find the probability density function that minimizes the integration process $P(x) \int_z P(z)^2 \mathrm{d}z\mathrm{d}x$. We change the order of integration and have Equation (26). Note that, we have the constraint $\int_x P(x) = 1 \int_z P(z) = 1$. Consequently, Equation (26) equals Equation (27). Afterward, Theorem 2 will solve Equation (27), finding the probability density function $P(x)$ that minimizes p_{attack}.

Theorem 2 (Optimal probability distribution). *When $P(z) \sim U$ (i.e., optimal distribution) and the probability of N_{config} device fingerprint configurations' occurrence is uniform, the insider attack has the smallest success probability. Besides, the larger the number of device fingerprint configurations, the lower the attack success probability.*

$$P_{min}^{attack} = \frac{1}{N_{config}}$$

Proof. We prove P_{min}^{attack} defined in Definition 8. We intend to get the probability density function $P(x)$ defined in Definition 9 that minimizes p_{attack}. Let the variable ac denotes the variation $\int_z P(z)^2$ in Definition 9, we can obtain Equation (28). Furthermore, we can derive Equation (29). According to Equation (36) in Theorem 3 (Jensen's inequality), we have Equation (30).

$$ac = \int_z P(z)^2 = \mathop{\mathrm{E}}_{z\sim P(z)} P(z) \tag{28}$$

$$\log ac = \log \mathop{\mathrm{E}}_{z\sim P(z)} P(z) \tag{29}$$

$$\log ac = \log \mathop{\mathrm{E}}_{z\sim P(z)} P(z) \geq \mathop{\mathrm{E}}_{z\sim P(z)} \log P(z) \tag{30}$$

Additionally, according to Theorem 4 (principle of maximum entropy), we have Equation (31).

$$\mathop{\mathrm{E}}_{z\sim P(z)} \log P(z) \geq \mathop{\mathrm{E}}_{z\sim U} \log U(z) \tag{31}$$

We can obtain the Equation (32) according to the inequality's transitive relation [4].

$$\log ac \geq \mathop{\mathrm{E}}_{z\sim P(z)} \log P(z) \geq \mathop{\mathrm{E}}_{z\sim U} \log U(z) \tag{32}$$

According to Theorem 4 (principle of maximum entropy), when $P(z) \sim U$, the variational lower bound $\mathrm{E}_{z\sim P(z)} \log P(z)$ obtains the minimum value. At this point $\log ac = \mathrm{E}_{z\sim P(z)} \log P(z)$ and $\mathrm{E}_{z\sim P(z)} \log P(z) = \mathrm{E}_{z\sim U} \log U(z)$. Consequently, $\log ac$ obtains the minimum value. Note that log is a monotonically increasing function, ac thus obtains the minimum value at this point. Then we can answer Q1 and Q2. When the probability distribution of device configuration is uniform, the insider attack success probability P_{min}^{attack} gets a minimum value.

The 3×3 sample grid in Fig. 10 simplifies P_{min}^{attack} calculation with discrete sample point selection probability. However, the actual sample point selection probability is continuous. Besides, the probability of an attacker selecting different sample points is the same as the probability of distinct device configurations' occurrence, i.e., $P_{sample} = P_{config}$. Because the attacker first guesses the device configuration and then computes the sample points. Each device configuration corresponds to a sample computation function. In the case of a sorting system, for a physical model with (v, h) selected, a m value must correspond to a s value, which means that (m, s) occurs with the same probability as (v, h). Therefore P_{min}^{attack} is minimized when the probability of each device fingerprint configurations' occurrence is equal to $\frac{1}{N_{config}}$ throughout the ICPS.

Equation (33) shows how to calculate P_{min}^{attack} with uniform distribution (i.e., optimal distribution $P(z) \sim U$). We have the simplified Equation (34) when the number of configurations N_{config} is relatively large.

$$\begin{aligned} P_{min}^{attack} &= \sum_{i=1}^{n} p_i \sum_{i\neq j}^{n} {p_j}^2 \\ &= [(n-1) \times \frac{1}{n^2}] \times n \times \frac{1}{n} = \frac{n-1}{n^2} \qquad (33) \\ &= \frac{1}{N_{config}} \quad s.t. \quad N_{config} \text{ is large} \qquad (34) \end{aligned}$$

□

Theorem 3 (Jensen's inequality). *For a convex function $f(x)$, numbers $x_1, x_2, x_3, ..., x_n \in [a, b]$, Jensen's inequality can be stated in Equation (35). The inequality is reversed if $f(x)$ is concave, which is Equation (36). Besides, equality holds if and only if $x_1 = x_2 = x_3 = ... = x_n$ [4].*

$$\frac{\sum_{i=1}^{n} f(x_i)}{n} \geq f(\frac{\sum_{i=1}^{n} x_i}{n}) \tag{35}$$

$$\frac{\sum_{i=1}^{n} f(x_i)}{n} \leq f(\frac{\sum_{i=1}^{n} x_i}{n}) \tag{36}$$

Theorem 4 (Principle of maximum entropy). *For a system with n events, the probability of each event occurring is p_i, such that the probability sum of the events occurring is 1. When the system entropy is maximum, the probability of each event is equal, i.e., $p_1 = p_2 = p_3 = \ldots = p_n$ and $P_i \sim U$ [14].*

$$H(P) = -\sum_{i=1}^{n} p_i \log p_i \quad s.t. \quad \sum_{i=1}^{n} p_i = 1 \tag{37}$$

Note that the proof of Theorem 2 uses Theorem 3 and Theorem 4.

Detection success probability. According to Theorem 2, we derive the minimum detection success probability in Equation (38). Specifically, if there are four velocity configurations and two height configurations in ICPS, the probability of the detection model correctly recognizing the random insider attacks is 87.5% ($1 - \frac{1}{4\times2}$). When the physical features identified for fingerprinting increase, the combination of physical features also boosts, leading to a rise in the total number of configurations in ICPS. Fig. 11 demonstrates that the proposed detection model thus will have a better performance in discovering insider attacks.

$$P_{max}^{detection} = 1 - P_{min}^{attack} = 1 - \frac{1}{N_{config}} \tag{38}$$

7 PERFORMANCE EVALUATION

In this section, we provide a set of comprehensive experiments. We propose multi-physical features based fingerprints that can easily classify device configurations and detect anomalies. The more features we identify, the better separability fingerprints have. Besides, the classifiers' accuracy, precision, and recall against the real-life dataset show that the detecting performance of classifiers approaches 100% when the device configuration number is small, demonstrating the effectiveness of fingerprinting methods. The confusion matrix further verifies the robustness of fingerprint generation.

7.1 Experimental Setup

Implementation. Fig. 12 shows the physical testbed that we build in the laboratory scenario. The testbed consists of a PC, controller, actuator, conveyor, velocity sensor, and camera, which simulate the sorting system in Fig. 3. Besides, the controller is the core control center. We use a microcontroller unit (MCU) as the controller and electrical machinery (motor) as the actuator. PC sends a command to the controller and receives the controller's response to the status of physical devices. The controller transmits the commands to the motor, connected to the motor and the velocity sensor. PC sends commands to the MCU to set the motor speed, thus changing the belt speed. When the belt is running, the velocity sensor in the testbed returns the current component velocity utilizing a raster calculation. The testbed adjusts the pulse width modulation (PWM) of the motor driver, which is connected between the MCU and the motor, changing the belt velocity. During the operation of the testbed, the experimental component will be thrown into different positions at the end of the conveyor belt. PC runs a python script using OpenCV to invoke the vision sensor to camera the experimental component. The python script performs contour extraction, calculates the pixel distribution of the component contour in the photograph, and then scales to calculate the distance to the component falling point and the size of the component cross-sectional area. PC stores all sensor data.

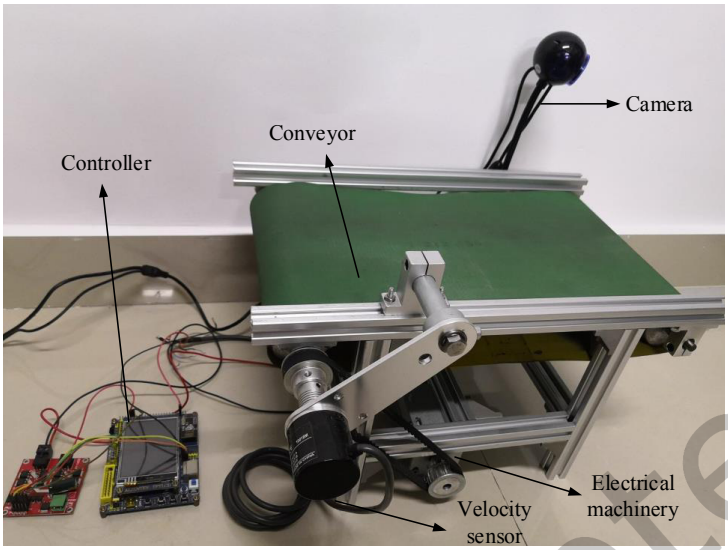


Fig. 12. Fingerprints generation through physical testbed

Datasets. In our experiment, the range of parameter v is ($0.8m/s$-$1.1m/s$), the range of m is ($5g$-$1000g$), and the h is $1.25m$. We adjust the working velocity into four levels by changing the PWM of the motor through the MCU. With the above settings, we can obtain fingerprints under various device configurations (physical characteristics). In §5.1, we propose to use real-life data collected from ICPS sensors for physical model training. We conduct up to 10,000 experiments to collect sensor data. Moreover, We remove the abnormal sensor feature data and eliminate interfering noise using the optimization methods in §5.3. We form a *dataset* with the sensor feature data, which contains physical properties such as velocity, mass, height, and displacements. We split the dataset into training and testing sets using the stratified K-fold method, where K is set to 10.

Evaluation metrics of classification effectiveness. The dataset collected in the real-life scenario may be unbalanced, i.e., there is a significant difference in the number of positive and negative category labels. Therefore a combination of *accuracy*, *precision*, and *recall* is required to evaluate the classification effectiveness of the unbalanced dataset. Each component in the dataset contains two labels: one is the label of its real category, and the other is the label given by the trained classifier. We can evaluate the classifier's effectiveness by comparing whether the two labels are consistent. Four situations exist for the comparison: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Equation (39) defines accuracy, precision and recall.

$$
\begin{aligned}
Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \\
Precision &= \frac{TP}{TP + FP} \\
Recall &= \frac{TP}{TP + FN}
\end{aligned}
\tag{39}
$$

7.2 Relationship Analysis of Multi-Physical Features

We analyze the relationships between the multiple physical features that construct a fingerprint by plotting the real-life sensor data. When visualizing the physical features, we found overlap between data points because the large amount of real-life data we have collected inevitably contains some abnormal data. In order to reduce the interfering noise data and visualize the data distribution better, we use the optimization method proposed in §5.3 and set the deviation σ_s as 5%. Fig. 13 shows the relationship between the component displacement and mass. The X-axis denotes the mass (Unit: g), and Y-axis is the component displacement (Unit: mm). The range of mass is from $5g$ to $1000g$. The conveyor belt height h is fixed to $1.25m$ and the velocity configuration contains four levels (v=$0.8m/s$, v=$0.9m/s$, v=$1.0m/s$, v=$1.1m/s$). Each fitted curve specifically represents the relationship between the optimal component displacement (s_y) and different masses (m_x) under one velocity level. We draw the fitted data point, the optimal displacement (s_y) and different masses (m_x), under one velocity level configuration. Then we fit the curve through univariate linear regression.

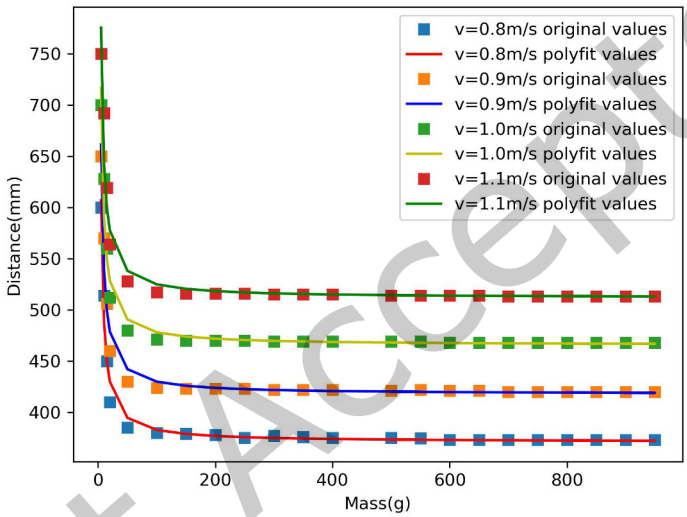


Fig. 13. The relationship between the component displacement and mass under different velocity levels

In the sorting system, we propose to use the fingerprint consisting of four physical features, mass, velocity, height, and displacement. Fig. 13 exactly visualizes the separability of device physical configurations based on their displacement measurements. Besides, it shows that the same configuration can generate stable fingerprints distinguishable from different configurations. From this physical phenomenon (Fig. 13 visualization), we can tell that even using simple metrics, such as displacement, results in the sorting line device configurations are highly separable.

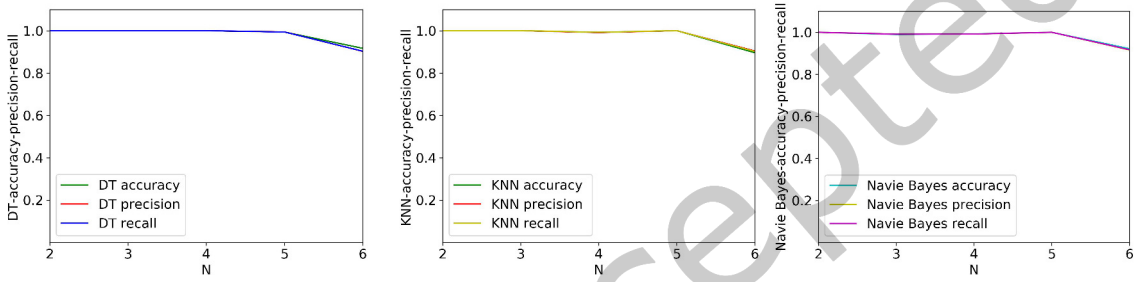
Also, we can easily observe that the greater the component mass, the smaller the displacement within a specific mass range. With the same mass, the greater the velocity, the greater the displacement. In a discrete manufacturing scenario, if the system sorts small components and component mass is not large, then the displacement distance is closely related to speed and mass. Additionally, we obtain the following two observations that are significant for fingerprint recognition methods from Fig. 13.

① **Classify device configurations based on their fingerprints**: The physical phenomenon suggests that a properly tuned machine learning **fingerprint classifier** model would result in high-accuracy device

configuration classification. Therefore, we propose to utilize a classifier in the differential detection model for fingerprint generation, comparing them with fingerprints received by sensors. If they are not identical, we can infer that an insider attack has happened and the attacker tampered with the sensor data. Specifically, we collect and optimize real featurized data from a physical testbed to train the **classifier** model.

② **A generalized classifier requires multi-physical features**: Fig. 13 shows that when the mass value is very small, there is an intersection of the fitted curves representing the relationship between mass and displacement under four-velocity configurations. This indicates that, in this case, it is impossible to distinguish the speed configurations by displacement and mass. In this situation, introducing other physical features, such as the conveyor belt height h, can change the intersection of the fitted curves. Consequently, we can conclude that a **generalized fingerprint classifier** requires multi-physical features.

7.3 Classifier Performance Analysis



(a) Decision Tree Classifier (b) KNN Classifier (c) Naive Bayes Classifier

Fig. 14. Performance of Decision Tree, KNN and Naive Bayes Classifiers

The fingerprint classification can distinguish the configuration type of a device. Furthermore, an effective classifier can detect false fingerprints generated by an attacker that does not belong to a particular configuration, thus **detecting insider attacks**. We build different kinds of classifiers to test their effectiveness. Intuitively, the more accurate the classifier is at classifying fingerprints correctly, the better it will be at detecting false fingerprints forged by attackers. We select three classification methods from supervised machine learning: Decision Tree (DT), K-Nearest Neighbors (KNN), and Naive Bayes (NB). We implement the classifiers in the Python Scikit-learn machine learning library.

For the testing dataset, we take the belt velocity v, height h, and the displacement s as physical features of component mass m. We can observe from Fig. 13 that when the component mass is too large, the component displacement depends on the height and velocity, independent of the mass. Therefore, we only used the velocity, height, and displacement features with smaller mass values (six categories of masses). Specifically, the classifier input is (v, h, s) and the classifier output is m. Different component masses indicate distinct component types. These four physical features form a fingerprint of a component.

Fig.14 (a)–(c) show the classification performance by DT, KNN and NB respectively. In each figure, X-axis denotes the number of classification types (N), i.e., the component type number that can be distinguished according to their physical features based fingerprints. Y-axis is the indicator of evaluating the classification effect. Three lines with different colors represent the accuracy, precision, and recall rate of the DT, KNN, and NB classifier.

Table 1 details the accuracy, precision, and recall data. We observe that when the component type number $N \leq 5$, the detection performance indicators are close to 100%. However, as N approaches 6, the classification

performance declines. We conclude from experiment 7.2, a generalized classification requires multi-physical features. As the classification types increases, we need to collect additional physical features to enhance the classifier performance. However, the industrial component sorting system sometimes only needs to distinguish a small number of component types. Therefore the ICPS configuration of (v, m, h, s) physical features is reasonable. Note that we benchmark three commonly used classifiers in the experiments. These classifiers have similar performance. Consequently, we can conclude that the classifier type does not affect the classification performance, and the classifiers are **robust**.

The confusion matrix analysis. We utilize the confusion matrix to further evaluate the classifier's performance in detecting insider attacks. Specifically, each column of the confusion matrix is the predicted category (component type), and the total number of columns represents the number of data predicted for that category. Each row is the actual category to which the data belongs, and the total amount of data in each row represents the number of data instances for that class. Moreover, to facilitate observation, we can divide the data of each cell by the total number and scale it. As shown in Fig. 15 (a) to (d), we conduct the confusion matrix analysis on the decision tree (DT) classifier, and the data on the diagonal is essentially 1. It means that the fingerprint classifier has better performance when N is 2 to 5. However, when N is 6, as shown in Fig. 15 (e), the detecting system may gain confusing results from the classifier. Because differentiating large-scale experimental components with relatively high mass *only by* velocity feature will produce an overlap of classification regions. When the ICPS sorts small components in discrete manufacturing industries, these situations do not affect the classification performance. We can get almost the same analysis results from the confusion matrix in KNN and NB. Consequently, this paper does not include the confusion matrix of the KNN and NB models.

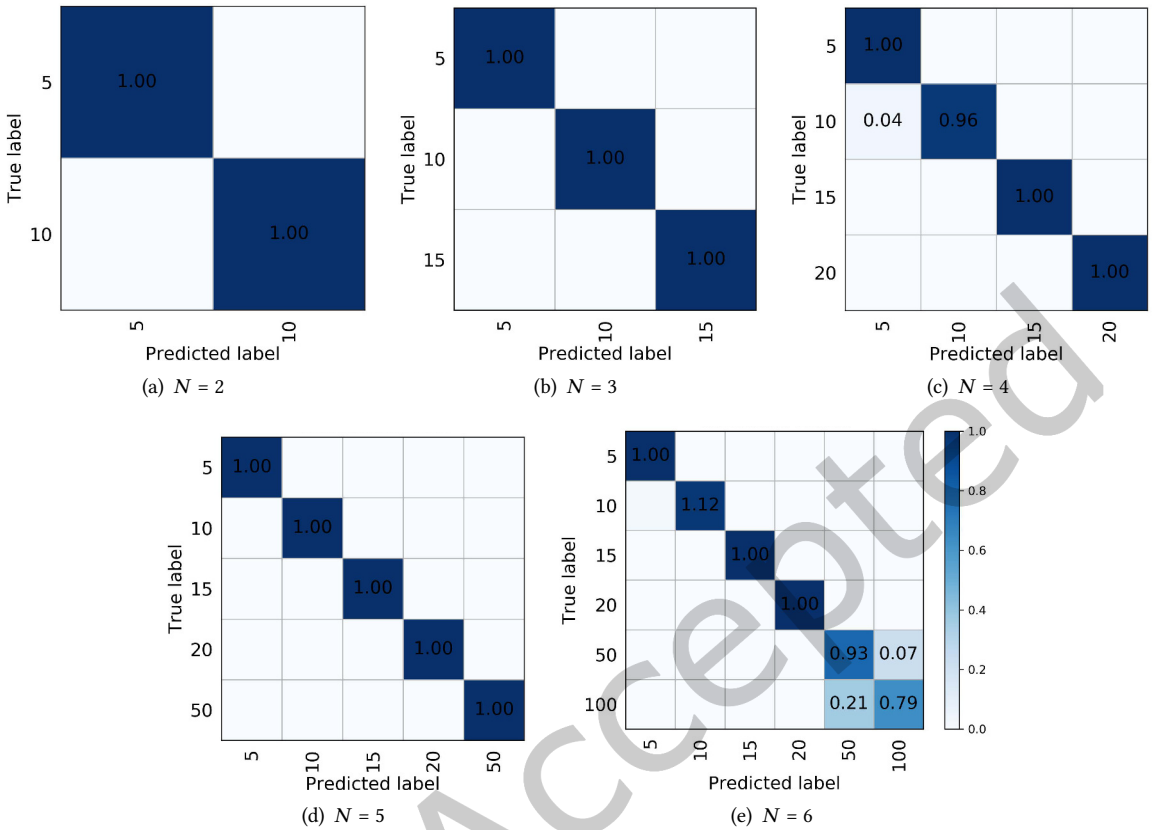
Detection success rate. We simulate the insider attack and use the r metric to calculate the classifier's detection success rate. We use a trained classifier to detect fingerprints forged by an attacker. The attacker randomly guesses the device configuration. As there are four sorting lines with different velocities in the ICPS, the attacker randomly chooses one of the four velocities to train the attack model. Experiment results show that the r metric of the decision tree classifier reaches 73.8%, indicating that the detection model discovers 73.8% attack behaviors. The r metric of the KNN classifier is 72.3%, and the Naive Bayes classifier's r metric is 73.4%. The r metric approaches the theoretical detection success probability ($1 - \frac{1}{4} = 75\%$) obtained via Equation (38) , demonstrating that the greater the physical device configurations number, the greater the probability that the detection model successfully discovers an attack.

8 CONCLUSION AND FUTURE WORK

In this paper, we propose a device fingerprinting technique based on multi-physical features to discover insidious insider attacks in ICPS. Notably, we demonstrate the fingerprinting approach and clarify the insider capabilities in the sorting system. Specifically, the fingerprint consists of four physical features (mass, velocity, height, and

Number of types	Decision Tree			KNN			Naive Bayes		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
2	100%	100%	100%	100%	100%	100%	100%	100%	100%
3	100%	100%	100%	100%	100%	100%	99.9%	99.2%	99.0%
4	99.2%	99.2%	99.3%	99.2%	99.2%	99.3%	100%	100%	100%
5	99.3%	99.3%	99.4%	99.3%	99.3%	99.4%	99.4%	99.4%	99.4%
6	88.5%	87.9%	87.8%	90.1%	90.5%	90.0%	91.7%	92.2%	91.5%

Table 1. The detailed accuracy, precision, and recall of Decision Tree, KNN, and Naive Bayes



(a) $N = 2$ (b) $N = 3$ (c) $N = 4$

(d) $N = 5$ (e) $N = 6$

Fig. 15. Decision tree confusion matrix with different category number N

displacement) that match relationships in the formalized physical model. Meanwhile, we implement a sorting testbed, collecting a large amount of feature data as a dataset. Utilizing featurized data, we train a classifier that generates fingerprints in real-time. Moreover, we employ a differential detection strategy based on device fingerprints to discover insider attacks efficiently. We solve the analysis of detection success probability as a variational problem, proving that the more device configurations, the lower the chance that an attacker successfully guesses the correct classifier's physical parameters. Experiments show that the more features we identify, the better the separability and detecting performance of fingerprints. Without loss of generality, the fingerprinting technique is robust and applies to intrusion detection of ICPS control flow in other applications.

For future work, we plan to construct device fingerprints based on the data acquisition and interaction functions of ICPS. Using the interaction protocols between multiple applications in cross-process manufacturing scenarios, we can implement the device fingerprinting approach of complex industrial control systems. Also, we can further use encryption technology in fingerprint synthesis, guarding the security and tamper-evident of fingerprints.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China under Grant 62072408, Zhejiang Provincial Natural Science Foundation of China under Grant LY20F020030, and New Century 151 Talent Project

of Zhejiang Province. We thank all of the anonymous reviewers who spent their own time reviewing and giving suggestions.

REFERENCES

[1] Abdallah Mustafa Abdelrahman, Joel JPC Rodrigues, Mukhtar ME Mahmoud, Kashif Saleem, Ashok Kumar Das, Valery Korotaev, and Sergei A Kozlov. 2021. Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions. *International Journal of Communication Systems* 34, 4 (2021), e4706.

[2] S. Abe, Y. Tanaka, Y. Uchida, and S. Horata. 2017. Tracking attack sources based on traceback honeypot for ICS network. In *Proceedings of 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE 2017)*. 717–723. https://doi.org/10.23919/SICE.2017.8105603

[3] C. M. Ahmed and A. P. Mathur. 2017. Hardware identification via sensor fingerprinting in a cyber physical system. In *Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C 2017)*. 517–524. https://doi.org/10.1109/QRS-C.2017.89

[4] Rabindra Nath Bhattacharya and Edward C Waymire. 2007. *A basic course in probability theory*. Vol. 69. Springer.

[5] Durbadal Chattaraj, Sourav Saha, Basudeb Bera, and Ashok Kumar Das. 2020. On the design of blockchain-based access control scheme for software defined networks. In *Proceedings of the 2020 International Conference on Computer Communications Workshops (NFOCOM Workshops)*. IEEE, 237–242.

[6] D. Ding, Q. Han, Y. Xiang, X. Ge, and X. Zhang. 2018. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing* 275 (Jan. 2018), 1674–1683. https://doi.org/10.1016/j.neucom.2017.10.009

[7] A. D'Innocenzo, F. Smarra, and M. Benedetto. 2016. Resilient stabilization of multi-hop control networks subject to malicious attacks. *Automatica* 71 (Sept. 2016), 1–9. https://doi.org/10.1016/j.automatica.2016.04.016

[8] Alessandro D'Innocenzo, Francesco Smarra, and Maria Domenica Di Benedetto. 2016. Resilient stabilization of multi-hop control networks subject to malicious attacks. *Automatica* 71 (2016), 1–9.

[9] David Formby, Preethi Srinivasan, Andrew M Leonard, Jonathan D Rogers, and Raheem A Beyah. 2016. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems.. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS)*. Internet Society.

[10] J. Francois, H. Abdelnur, R. State, and O. Festor. 2011. PTF: passive temporal fingerprinting. In *Proceedings of 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. 289–296. https://doi.org/10.1109/INM.2011.5990703

[11] Thomas Fulton and Fritz Rohrlich. 1960. Classical radiation from a uniformly accelerated charge. *Annals of Physics* 9, 4 (1960), 499–517.

[12] Q. Gu, D. Formby, S. Ji, H. Cam, and R. Beyah. 2018. Fingerprinting for cyber-physical system security: device physics matters too. *IEEE Security & Privacy* 16, 5 (Sept. 2018), 49–59. https://doi.org/10.1109/MSP.2018.3761722

[13] Zhen Hong, Chaofeng Yang, and Li Yu. 2020. *R*-Print: A System Residuals-Based Fingerprinting for Attack Detection in Industrial Cyber-Physical Systems. *IEEE Transactions on Industrial Electronics* 68, 11 (2020), 11458–11469.

[14] Jagat Narain Kapur. 1989. *Maximum-entropy models in science and engineering*. John Wiley & Sons.

[15] Abdelaziz Khaled, Samir Ouchani, Zahir Tari, and Khalil Drira. 2020. Assessing the severity of smart attacks in industrial cyber-physical systems. *ACM Transactions on Cyber-Physical Systems* 5, 1 (2020), 1–28.

[16] David A Knox and Thomas Kunz. 2015. Wireless fingerprints inside a wireless sensor network. *ACM Transactions on Sensor Networks (TOSN)* 11, 2 (2015), 1–30.

[17] R. Langner. 2011. Stuxnet: dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (May 2011), 49–51. https://doi.org/10.1109/MSP.2011.67

[18] Yuxiang Lin, Yi Gao, Bingji Li, and Wei Dong. 2022. Detecting Rogue Access Points Using Client-agnostic Wireless Fingerprints. *ACM Transactions on Sensor Networks (TOSN)* 19, 1 (2022), 1–25.

[19] J. Lopez, N. C. Liefer, C. R. Busho, and M. A. Temple. 2018. Enhancing critical infrastructure and key resources (CIKR) level-0 physical process security using field device distinct native attribute features. *IEEE Transactions on Information Forensics and Security (TIFS)* 13, 5 (May 2018), 1215–1229. https://doi.org/10.1109/TIFS.2017.2779447

[20] Lingling Lu, Jianhai Chen, Zijun Tian, Qinming He, Butian Huang, Yang Xiang, and Zhenguang Liu. 2019. EduCoin: a Secure and Efficient Payment Solution for MOOC Environment. In *Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain 2019)*. IEEE, 490–495.

[21] Chuck McParland, Sean Peisert, and Anna Scaglione. 2014. Monitoring Security of Networked Control Systems: It's the Physics. *IEEE Security & Privacy* 12, 6 (2014), 32–39.

[22] C. Neilson. 2013. Securing a control systems network. *ASHRAE Journal* 55, 11 (2013), b18–b22.

[23] A. Nourian and S. Madnick. 2018. A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet. *IEEE Transactions on Dependable and Secure Computing (TDSC)* 15, 1 (Jan.-Feb. 2018), 2–13. https://doi.org/10.1109/TDSC.2015.2509994

[24] K. Paridari, N. O'Mahony, A. E. D. Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg. 2018. A framework for attack-resilient industrial control systems: attack detection and controller reconfiguration. *Proceedings of the IEEE (Proc. IEEE)* 106, 1 (Jan. 2018), 113–128. https://doi.org/10.1109/JPROC.2017.2725482

[25] Christian W Probst, René Rydhof Hansen, and Flemming Nielson. 2006. Where can an insider attack?. In *International Workshop on Formal Aspects in Security and Trust*. Springer, 127–142.

[26] D. R. Reising, M. A. Temple, and J. A. Jackson. 2015. Authorized and rogue device discrimination using dimensionally reduced RF-DNA fingerprints. *IEEE Transactions on Information Forensics and Security (TIFS)* 10, 6 (Jun. 2015), 1180–1192. https://doi.org/10.1109/TIFS.2015.2400426

[27] Lior Rokach and Oded Maimon. 2005. Decision trees. In *Data mining and knowledge discovery handbook*. Springer, 165–192.

[28] S. Sridhar, A. Hahn, and M. Govindarsu. 2012. Cyber-physical system security for the electric power grid. *Proc. IEEE* 100, 1 (Jan. 2012), 210–224. https://doi.org/10.1109/JPROC.2011.2165269

[29] Anil Kumar Sutrala, Mohammad S Obaidat, Sourav Saha, Ashok Kumar Das, Mamoun Alazab, and Youngho Park. 2021. Authenticated key agreement scheme with user anonymity and untraceability for 5g-enabled softwarized industrial cyber-physical systems. *IEEE Transactions on Intelligent Transportation Systems (TITS)* 23, 3 (2021), 2316–2330.

[30] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. 2016. Limiting the impact of stealthy attacks on industrial control systems. In *2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1092–1105. https://doi.org/10.1145/2976749.2978388

[31] Youngseok Yang, Taesoo Kim, and Byung-Gon Chun. 2021. Finding consensus bugs in ethereum via multi-transaction differential fuzzing. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2021)*. 349–365.

[32] Y. Yang, K. Mclaughlin, T. Littler, S. Sezer, and H. F. Wang. 2014. Rule-based intrusion detection system for SCADA networks. In *Proceedings of 2nd IET Renewable Power Generation Conference*. https://doi.org/10.1049/cp.2013.1729

[33] W. Zhang, Z. Hong, J. Zhu, and B. Chen. 2019. A survey of network intrusion detection methods for industrial control systems. *Control and Decision* 34, 11 (Nov. 2019), 2277–2288. https://doi.org/10.13195/j.kzyjc.2019.1302