Hindawi
Security and Communication Networks
Volume 2023, Article ID 7010155, 12 pages
https://doi.org/10.1155/2023/7010155

Research Article

Privacy-Preserving Industrial Control System Anomaly Detection Platform

Shan Gao,[1] Junjie Chen,[1,2] Bingsheng Zhang,[1] and Kui Ren[1,3]

[1]*School of Cyber Science and Technology, Zhejiang University, Hangzhou, China*
[2]*Hangzhou Global Scientific and Technological Innovation Center, Zhejiang University, Hangzhou, China*
[3]*Zhejiang Provincial Key Laboratory of Blockchain and Cyberspace Governance, Hangzhou, China*

Correspondence should be addressed to Junjie Chen; 22121095@zju.edu.cn

Received 26 May 2022; Revised 9 July 2022; Accepted 24 November 2022; Published 28 April 2023

Academic Editor: Clemente Galdi

Copyright © 2023 Shan Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of IT technologies, an increasing number of industrial control systems (ICSs) can be accessed from the public Internet (with authentication). In such an open environment, cyberattacks become a serious threat to both ICS system integrity and data privacy. As a countermeasure, anomaly detection systems are often deployed to analyze the network traffic. However, due to privacy regulation, the network packages cannot be directly processed in plaintext in many countries. In this work, we present a privacy-preserving anomaly detection platform for ICS. The platform consists of three nodes running low-latency MPC protocols to evaluate the live network packages using decision trees on the fly with privacy assurance. Our benchmark result shows that the platform can process thousands of packages every ten seconds.

1. Introduction

A modern industrial control system (ICS) is a complex distributed system that consists of multiple field devices, e.g., sensors, actuators, and instrumentation, as well as some control/management systems. ICS is the interface of cyber-physical system (CPS), enabling humans to control operations and receive data from devices. In recent years, ICS has been widely used in many industrial scenarios, such as gas, water, and nuclear power systems, and the security of these systems is critical.

As shown in Figure 1, a typical architecture of an industrial control system has four layers. (i) The enterprise management layer offers business services and is often connected to public network, which may include the enterprise resource planning (ERP) system, manufacturing execution system (MES), and management information system (MIS). (ii) The supervisory control layer receives and stores data from the underlying devices and then gives appropriate responses. (iii) The process control layer has programmable logic controller (PLC) and remote terminal unit (RTU), which directly control devices in the underlying layer. (iv) The field control layer has multiple field devices that receive commands and send data to the process control layer. As the enterprise management layer connects to the public network, ICS is exposed to cyberattacks. Along with the advancement of cyberattacks, the corresponding countermeasure techniques also need to be upgraded. In practice, a great number of famous industrial control systems have been severely threatened by cyberattack. For instance, the Stuxnet virus spied and reprogrammed industrial systems controlling centrifuges of the Iran nuclear power plant [1]. In 2021, hackers breached Colonial Pipeline using compromised password and Colonial Pipeline had to give hackers ransom [2].

To enhance industrial control system security, defense systems like intrusion (or anomaly) detection system (IDS) are deployed in ICS. IDS plays an important role in protecting ICS, which is commonly used to detect potential cyberattacks. IDS can be classified as network intrusion detection system (NIDS) and host-based intrusion detection system (HIDS). The NIDS examines network traffic, while the HIDS monitors the system data logs. According to detection approach, IDS can be classified as signature-based

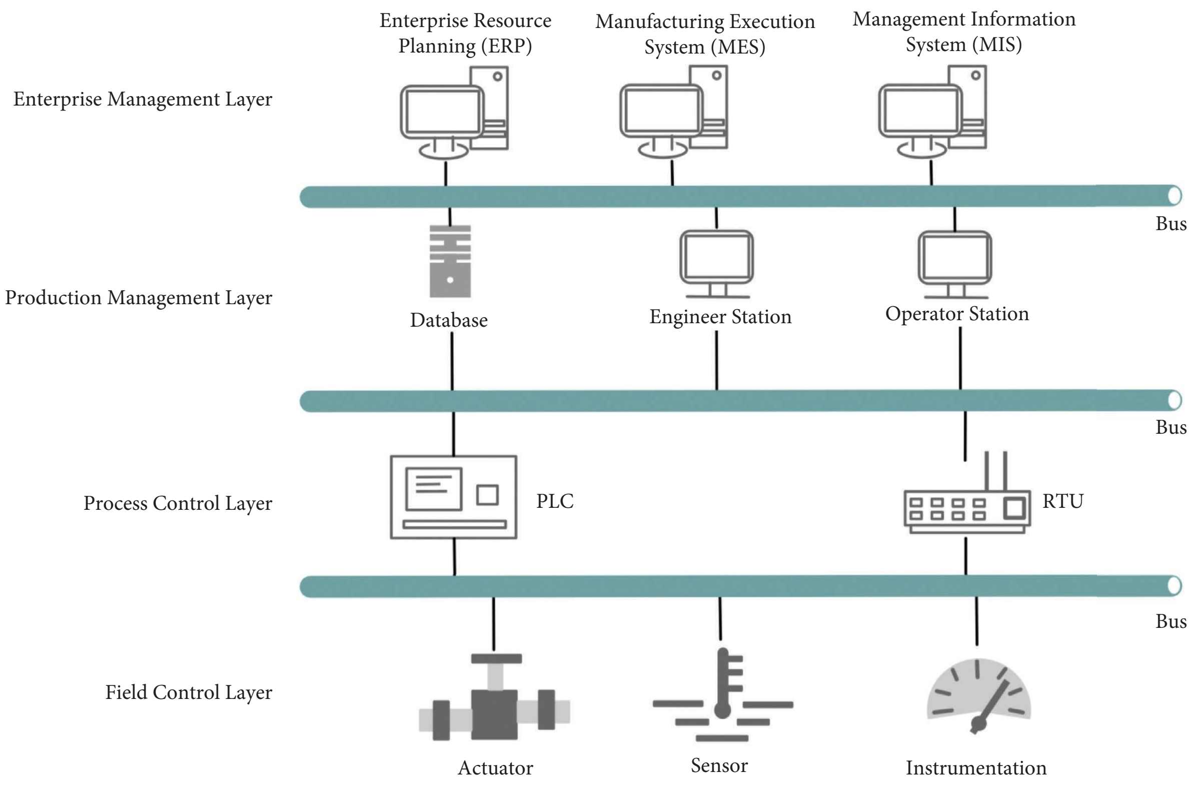
Enterprise Resource Planning (ERP)
Manufacturing Execution System (MES)
Management Information System (MIS)
Enterprise Management Layer
Bus
Production Management Layer
Database
Engineer Station
Operator Station
Bus
Process Control Layer
PLC
RTU
Bus
Field Control Layer
Actuator
Sensor
Instrumentation

FIGURE 1: A typical architecture of industrial control system.

detection and anomaly-based detection. The former detects intrusion by recognizing harmful system pattern, while the latter does it by analyzing network traffic packages.

In this work, we aim to design an anomaly-based network intrusion detection platform for ICS. The platform can be deployed alongside any existing off-the-shelf ICSs, and it can examine live network packages on the fly and raise alarms once fault is detected. However, in many countries, processing network packages in plaintext violates the local privacy laws and regulations. The European Union has put forward *General Data Protection Regulation* [3] in 2016, which has a clear standard for the processing of personal information. In 2020, The United States carried out the *California Consumer Privacy Act* [4], creating a series of privacy rights for consumers, such as the right to access, delete, and know. In China, new 2020 edition of the *Personal Information Security Specification* [5] has proposed clear regulations in the life cycle of the personal information, including collection, storage, use, processing, transmission, openness, and deletion. These regulations will have a profound impact to systems that store and/or process personal information. Therefore, our anomaly detection platform is designed to be privacy preserving.

As a closely related work, Gao et al. [6] used the homomorphic encryption scheme to encrypt data when training and applying ICS-specific anomaly detection model. But homomorphic encryption scheme will lead to heavy computation overhead. Alternatively, we utilize low-latency secure multi-party computation (MPC) techniques for privacy-preserving anomaly detection.

More specifically, our platform consists of three non-colluding servers that run low-latency MPC protocols to analyze network package in real time using the gradient boosting decision tree (GBDT) model with privacy assurance. GBDT is an effective machine learning algorithm which classifies input data rapidly with high accuracy.

1.1. Our Contributions.

In this work, we present an efficient MPC-based privacy-preserving anomaly detection platform for ICS. More specifically, the contributions of this work are as follows:

(i) We propose a new MPC-based anomaly detection architecture for ICS, and it is compatible with any off-the-shelf ICSs.

(ii) We design several new constant-round low-latency MPC protocols for privacy-preserving decision tree evaluation.

(iii) We implement a prototype of the proposed system, and our benchmark result shows that processing 1000 network packages with a depth-9 decision tree takes 11 seconds in the LAN setting.

1.2. Roadmap.

The remainder of this paper is organized as follows. We introduce the preliminary knowledge about the approach we used in Section 2. Then, system overview and security model of our platform are given in Section 3. Section 4 describes privacy-preserving decision tree evaluation in detail. We present the performance of the proposed platform in Section 5. The related work is given in Section 6. Finally, conclusion and future work are given in Section 7.

2. Preliminary

2.1. Notations. Throughout the paper, we use the following notations. Denote τ as the security parameter. Denote a value x indexed by a label i as $(x)_i$. (s,t)-secret sharing is to divide secret into t parts, and any s participants can reveal secret jointly. Denote (2, 2)-additive secret sharing, (3, 3)-additive secret sharing, and (2, 3)-additive secret sharing in $\mathbb{Z}_n$ in Table 1.

$r \longleftarrow R$ means to randomly sample the element r from the set R. In addition, $y \longleftarrow f(x)$ represents y is the output when the function $f()$ takes x as input. For $x \in [-2^{\ell-1}, 2^{\ell-1}]$, map it to $\mathbb{Z}_{2^\ell}$ by adding $2^{\ell-1}$.

2.2. Gradient Boosting Decision Tree. Our proposed platform mainly uses gradient boosting decision tree (GBDT) as intrusion detection model. Decision tree is a classical machine learning model, which is efficient and interpretable. Its non-leaf node is decision node, which performs a test to decide to go to left sub-tree or right sub-tree. Its leaf node is the end of a decision path that begins with root node, including prediction result. Boosting is a kind of algorithm that combines many weak learners into a strong learner. The first step is training a base learner, like decision tree. Then, adjust training samples according to the classification result of base learner, so that those misclassified samples will get more attention in the subsequent training process. After that, train next weak learner using adjusted training samples. Repeat the process iteratively to obtain enough weak classifiers and combine them together according to their weight to obtain a strong classifier. Gradient boosting is an algorithm in boosting, which iterates the new learner through gradient descent.

The GBDT is a learning algorithm based on boosting. Its essence is that the next regression decision tree is built on the gradient descent direction of the loss function of the last round, and multiple regression decision trees are combined into a gradient boosting decision tree finally. When x is the input of GBDT, its classification result is $\widehat{y} = \sum_{k=1}^{K} f_k(x)$, where K is number of decision trees in GBDT and $f_k(x)$ is k-th tree's output. In general, the tree in GBDT is CART tree.

Given a training set $S: = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is input feature vector and y_i is its class label. The process of training GBDT consists of T rounds iteration. In the k-th iteration, the goal is to generate a decision tree f_k to minimize the objective function L.

$$L^{(k)} = \sum_{i=1}^{n} l\left(y_i, \widehat{y}_i^{(k-1)} + f_k(\mathbf{x}_i)\right) + \Omega(f_k), \tag{1}$$

where $\Omega(f) = \gamma T + 1/2\lambda \sum_{j=1}^{T} \omega_j^2$ is regularization item and $\widehat{y}_i^{(k-1)}$ is i-th sample's classification result in k-th iteration. l is loss function, T is number of leaf nodes, and ω_j is the value of a leaf node. Expression (1) uses a second-order Taylor expansion to get the following expression.

$$L^{(k)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \widehat{y}^{(k-1)}\right) + g_i f_k(\mathbf{x}_i) + \frac{1}{2} h_i f_k^2(\mathbf{x}_i)\right] + \Omega(f_k), \tag{2}$$

where $g_i = \partial_{\widehat{y}^{(k-1)}} l(y_i, \widehat{y}^{(k-1)})$ is the first step degree value of loss function l and $h_i = \partial^2_{\widehat{y}^{(k-1)}} l(y_i, \widehat{y}^{(k-1)})$ is the second step degree value of loss function l.

The k-th decision tree only includes a root node with the training set S initially. Suppose a sample set S in a node is partitioned into S_l and S_r; $L_{\text{split}}(S_l, S_r)$ is defined as follows.

$$L_{\text{split}}(S_l, S_r) = -\frac{1}{2}\left[\frac{\left(\sum_{i\in S_l} g_i\right)^2}{\sum_{i\in S_l} h_i + \lambda} + \frac{\left(\sum_{i\in S_r} g_i\right)^2}{\sum_{i\in S_r} h_i + \lambda}\right] + \gamma T. \tag{3}$$

Perform a test for each possible split point and select the optimal split point that causes minimum L_{split}. If the current node does not meet the splitting requirements, for example, the depth of the current node reaches the maximum, the current node becomes a leaf node with a value $V(S)$, and $V(S)$ is defined as follows.

$$V(S) = -\frac{\sum_{i\in S} g_i}{\sum_{i\in S} h_i + \lambda}. \tag{4}$$

GBDT has strong classification ability in anomaly detection task.

2.3. Secure Multi-Party Computation. Secure multi-party computation permits two or more participating parties to obtain output result by jointly computing over sensitive data from respective inputs. At the same time, the participating parties do not learn more about other parties' inputs than the information about the output, so that each participating party can get computation result without leaking sensitive message.

Secure multi-party computing usually includes two different adversary models, namely, semi-honest security model and malicious security model. A semi-honest security model is one in which the adversary will honestly perform the intended calculation process but may wish to know the information of each party to the maximum extent. A malicious security model is one in which an adversary can control, manipulate, and arbitrarily contaminate information on a multi-party computing network. In this work, we mainly consider the semi-honest security model.

Although the first MPC protocol was already proposed by A. C.-C. Yao [7] in the 1980s, it was implemented practically in the last eighteen years. Nowadays, MPC becomes more important as data privacy gets more and more attention. It was adopted for private set intersection [8] and privacy-preserving machine learning [9].

Table 1: (s, t)-secret sharing.

(s,t)-additive sharing in $\mathbb{Z}_n$	Description
(2, 2)-additive sharing	$[x] := \{(x)_0, (x)_1\}$, where $x = (x)_0 + (x)_1 \pmod{n}$. S_0 holds $(x)_0$ and S_1 holds $(x)_1$
(3, 3)-additive sharing	$\langle x\rangle := \{(x)_0, (x)_1, (x)_2\}$, where $x = (x)_0 + (x)_1 + (x)_2 \pmod{n}$. S_0 holds $(x)_0$, S_1 holds $(x)_1$, and S_2 holds $(x)_2$
(2, 3)-replicated sharing	$\langle x\rangle^r := \{(x)_0, (x)_1, (x)_2\}$, where $x = (x)_0 + (x)_1 + (x)_2 \pmod{n}$. S_0 holds $\{(x)_0, (x)_1\}$, S_1 holds $\{(x)_1, (x)_2\}$, and S_2 holds $\{(x)_2, (x)_0\}$

Secret sharing is one of important parts in MPC. The remaining part of this section introduces distributed interval containment function (DICF) and shared oblivious transfer that we adopt in our MPC protocol.

2.3.1. Function Secret Sharing. Function secret sharing (FSS) [10] can split a function $f: \mathbb{X} \longrightarrow \mathbb{Y}$ into $\{(f)_0: \mathbb{X} \longrightarrow \mathbb{Y}, (f)_1: \mathbb{X} \longrightarrow \mathbb{Y}\}$ and $f(x) = (f(x))_0 + (f(x))_1 \pmod{|\mathbb{Y}|}$ for each $x \in \mathbb{X}$, where $|\mathbb{Y}|$ denotes the number of element in $\mathbb{Y}$. Distributed point function (DPF) is a FSS scheme. For a point function $f_{\alpha,\beta}(x): \mathbb{X} \longrightarrow \mathbb{Y}$, the range $\mathbb{Y}$ has only one non-zero value $f_{\alpha,\beta}(\alpha) = \beta$. There are two algorithms in DPF:

(i) $\mathrm{Gen}(1^\lambda, f_{\alpha,\beta})$: It generates a pair of keys $((\mathscr{F})_0 (\mathscr{F})_1)$. Each key is the share of $f_{\alpha,\beta}$ without revealing α and β.

(ii) $\mathrm{Eval}(i, (\mathscr{F})_i, x)$: $\forall x \in \mathbb{X}$, it outputs $(\beta_x)_i$, such that $(\beta_x)_0 + (\beta_x)_1 = f_{\alpha,\beta}(x) \pmod{|\mathbb{Y}|}$.

Denote run Eval on all inputs by $\mathrm{EvalAll}(i, (\mathscr{F})_i)$.

DICF [11] is also a FSS scheme that can judge whether a secret input value is in a publicly known interval. Denote an interval containment function as the following equation.

$$\mathbf{F}_{p,q}(x) = \begin{cases} 1, & if\ x \in [p,q], \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

DICF uses offset interval containment function defined as follows.

$$\mathbf{F}_{p,q,r_{\mathrm{in}},r_{\mathrm{out}}}(x + r_{\mathrm{in}}) = \mathbf{F}_{p,q}(x) + r_{\mathrm{out}}, \tag{6}$$

where r_{in} and r_{out} are random offset values. Like DPF, DICF also consists of two algorithms.

(i) $\mathrm{Gen}(1^\lambda, \mathbf{F}_{p,q,r_{\mathrm{in}},r_{\mathrm{out}}})$: it generates $((\mathscr{F})_0, (\mathscr{F})_1)$, as p, q are publicly known and r_{in} and r_{out} are unrevealed.

(ii) $\mathrm{Eval}(i, (\mathscr{F})_i, x + r_{\mathrm{in}})$: outputs a result $(\beta)_i$, so that $(\beta)_0 + (\beta)_1 - r_{\mathrm{out}} = \mathbf{F}_{p,q}(x) \pmod{|\mathbb{Y}|}$.

2.3.2. Oblivious Transfer. Oblivious transfer (OT) [12] is an important basic block in many MPC protocols. In oblivious transfer protocol, a sender has multiple messages and only one of them will be selected by receiver. Which message is selected is oblivious to the sender and the receiver can only obtain the selected message.

Shared OT is a kind of OT scheme that is used to fetch value in the shared form without revealing the value. In our approach, we utilize a 3-party shared OT protocol. In this protocol, three participants S_0, S_1, S_2 share a data vector $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_{n-1})$ and an index i ($i \in \mathbb{Z}_n$), as S_0 holds $\{(\mathbf{x})_0, (\mathbf{x})_1, (i)_0\}$, S_1 holds $\{(\mathbf{x})_1, (\mathbf{x})_2, (i)_1\}$, and S_2 holds $\{(\mathbf{x})_2, (\mathbf{x})_0, (i)_2\}$, where $(x)_j = ((x_0)_j, (x_1)_j, (x_2)_j, \ldots, ((x_{n-1})_j))$. Then, they can fetch x_i in the shared form without revealing i by jointly computing.

2.4. Intrusion Detection. The main goal of intrusion detection system [13] is to detect cyberattacks. Cyberattack is any type of offensive action against computer systems, computer networks, or personal computer. Damaging, exposing, modifying, disabling software or services, or stealing or accessing data from any computer without authorization is considered an attack on the computer and computer network. According to the attack mode, cyberattack can be divided into active attack and passive attack. An active attack attempts to destroy computer system, which includes denial of service (DoS), distributed denial of service (DDoS), and botnet, while a passive attack aims to learn information about network system like port scan attack.

DoS deliberately attacks flaws in the network protocol implementation or depletes the target's resources by brutal means, so that service or network cannot provide normal services. DDoS is a special form of denial of service attack based on DoS. It is a distributed and coordinated large-scale attack that may come from multiple attackers.

Botnet refers to the use of one or more means of transmission to infect a large number of hosts with bot program virus, so as to form a one-to-many control network between the controller and the infected host. The attacking process of port scanning attack is usually to remotely scan each port of the target computer, detect the services provided by different ports, and then record the response of the target computer to collect its information.

Generally, network anomaly detection requires the information about data packets, such as packet header characteristics, characteristics about TLS, and packet length.

3. System Framework and Security Model

This section gives the overview of our system framework firstly and describes the security model in Section 3.2.

3.1. System Framework. There are several components in the system framework, as depicted in Figure 2. The ICS preprocessed its packages firstly, including extracting features and secret sharing. For each data package, a feature vector is extracted from it. The feature vector contains the information about packet header and packet length. Then, the feature vector is divided into three parts using (2, 3)-additive secret sharing among S_0, S_1, S_2. Next, the parts of secret are distributed to three servers, where a well-trained CART model is stored in the shared form using (2, 2)-additive secret sharing between S_0 and S_1. Finally, the three nodes jointly obtain a detection result based on CART model, running MPC protocols described in Section 4.

3.2. Security Model. In the process of anomaly detection, we cannot store and process sensitive data from ICS in plaintext, according to respective laws and regulations. In order to detect attacks in network traffic with privacy assurance, we adopt MPC to achieve our goals. Firstly, we assume that there is a component in the ICS that can extract feature vectors of its network packages. This component shall be trusted, and it will then secretly share the extracted features to the three MPC nodes of our platform. One out of the three MPC nodes can be semi-honestly corrupted by the adversary. The shared process result will be sent back to the system admin of ICS, who will recover the result and make further actions accordingly.

3.2.1. Security Requirements. As described above, our proposed platform should protect privacy of ICS data when examining the sensitive data. Besides, the platform should respond accurately and quickly so that ICS can identify anomalies in time. Thus, we define the following key security requirements.

(i) *Data Privacy.* Though we detect the sensitive data from ICS, the data will not be stored or processed in plaintext. Even if a MPC node is semi-honestly corrupted by the adversary, the data privacy can still be protected.

(ii) *Accuracy.* As the platform's task is anomaly detection, the accuracy of detection model should be as high as possible.

(iii) *On Time.* Our platform should respond ICS as fast as possible so that ICS can handle cyberattack timely.

4. Privacy-Preserving Decision Tree Evaluation

This section describes the MPC protocols utilized in our approach. Firstly, we describe 3-party shared OT in Section 4.1. Then, we give the whole detection process, including data preprocessing, tree model storage, and evaluation.

4.1. 3-Party Shared OT. Given a replicated shared data vector $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ and an additively shared index $i \in \mathbb{Z}_n$, three participants hold the shared form $\{(\mathbf{x})_0, (\mathbf{x})_1, (i)_0\}$, $\{(\mathbf{x})_1, (\mathbf{x})_2, (i)_1\}$, and $\{(\mathbf{x})_2, (\mathbf{x})_0, (i)_2\}$ respectively, that is similar to [14]. Then, the three participants can obtain x_i in shared form by running our 3-party shared OT protocol.

4.1.1. Intuition. Our protocol is mainly constructed on the basis of Paul et al. [14], whose main idea is that each participant serves as the generator of DPF scheme, while the other two participants serve as evaluators to get i-th value of vector $\mathbf{x}$ in the shared form. For instance, let S_0 be the DPF generator and S_1, S_2 be the DPF evaluators. Firstly, S_1 and S_2 randomly select $r_1, r_2 \longleftarrow \mathbb{Z}_n$, respectively. Then, S_1, S_2 exchange $r_1 - (i)_1, r_2 - (i)_2$ and send r_1, r_2 to S_0. After that, S_1, S_2 compute $\sigma\colon = r_1 - (i)_1 + r_2 - (i)_2 \pmod n$ and S_0 computes $\theta\colon = (i)_0 + r_1 + r_2 \pmod n$. It is easy to see that $\theta - \sigma = i \pmod n$. Next, S_0 generates a pair of DPF keys for point function $f_{\theta,1}(x)$ and sends keys to evaluators. Finally, S_1, S_2 run full domain evaluation to jointly obtain $\{[\beta_{0,\theta}], [\beta_{1,\theta}], \ldots, [\beta_{n-1,\theta}]\}$ ($[\beta_{k,\theta}]$ is 1 if $k = \theta$, and is 0 otherwise). Note that θ-th element in shifted vector is $[(x_i)_2]$, as $[(\mathbf{x})_2]$ is cyclic shifted to the right σ position. After all these steps, S_1, S_2 hold $[(x_i)_2]$ in shared form. Following similar steps, S_0, S_2 can jointly get $[(x_i)_0]$ and S_0, S_1 can jointly get $[(x_i)_1]$. In our 3-party shared OT protocol, we let the generator produce DPF keys of $f_{\mu,1}(x)$, where $\mu \longleftarrow \mathbb{Z}_n$ is randomly picked by generator. Then, the generator can produce DPF keys, which leads to less communication. Subsequently, all participants jointly compute and reveal $\langle\sigma\rangle\colon = \langle i\rangle + \langle 0\rangle - \mu$ to evaluators. At the end, evaluators can get $[(x_i)_0], [(x_i)_1], [(x_i)_2]$ in the shared form.

4.1.2. Protocol Description. The 3-party shared OT is depicted in Protocol 1. Initially, for $j \in \mathbb{Z}_3$, S_j and S_{j+1} agree on a random seed $\varphi_j \in \{0,1\}^\tau$ as $j \in \mathbb{Z}_3$, and if index $(j+1)$ greater than 2, S_{j+1} is the brief form of $S_{j+1 \pmod 3}$. Note that as the index $j \in \mathbb{Z}_3$, we omit (mod 3) in the rest of this paper. Before each round of shared OT, for $j \in \mathbb{Z}_3$, node S_j generates DPF keys $((\mathscr{F}_{\mu_j})_0, ((\mathscr{F}_{\mu_j})_1)$. Then, S_j sends $(\mathscr{F}_{\mu_j})_0$ to S_{j+1} and $(\mathscr{F}_{\mu_j})_1$ to S_{j+2}. All participants generate some $\langle 0\rangle$ using random seeds and pseudo-random function PRF. They jointly compute and reveal $\langle\sigma_j\rangle := \langle i\rangle + \langle 0\rangle - \mu_j \pmod N$ to evaluators S_{j+1} and S_{j+2}. Next, evaluators S_{j+1}, S_{j+2} use DPF keys $((\mathscr{F}_{\mu_j})_0, ((\mathscr{F}_{\mu_j})_1)$ to get $\{[\beta_{0,\mu_j}], [\beta_{1,\mu_j}], \ldots, [\beta_{n-1,\mu_j}]\}$ by running EvalAll algorithm. Then, they jointly obtain

$$[(x_i)_{j+2}] = \sum_{k=0}^{n-1}\left(\left(x_{k+\sigma_j}\right)_{j+2} \bullet \left[\beta_{k,\mu_j}\right]\right)\left(\mathrm{mod}\, 2^{\ell}\right). \tag{7}$$

They can jointly get x_i as $x_i = (x_i)_0 + (x_i)_1 + (x_i)_2$. Lastly, participants rerandomize shares to ensure their uniform distribution.

4.2. Data Preprocessing. Before ICS transmits its network packages, the data need to be preprocessed in two steps. Firstly, ICS extracts a feature vector for each package so that decision tree can detect on package level. Then, it completes data desensitization. A feature vector $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ is shared as $\langle x\rangle^r = \{(\mathbf{x})_0, (\mathbf{x})_1, (\mathbf{x})_2\}$, where $(x)_j = ((x_0)_j, (x_1)_j, \ldots, (x_{n-1})_j)$ and $j \in \mathbb{Z}_3$. Then S_0 holds $\{(\mathbf{x})_0, (\mathbf{x})_1\}$, S_1 holds $\{(\mathbf{x})_1, (\mathbf{x})_2\}$, and S_1 holds $\{(\mathbf{x})_2, (\mathbf{x})_0\}$.

4.3. Storage of Tree Model. As we adopt a constant-round MPC protocol that needs a full binary tree and our trained model is just a binary tree, we will pad the binary tree as

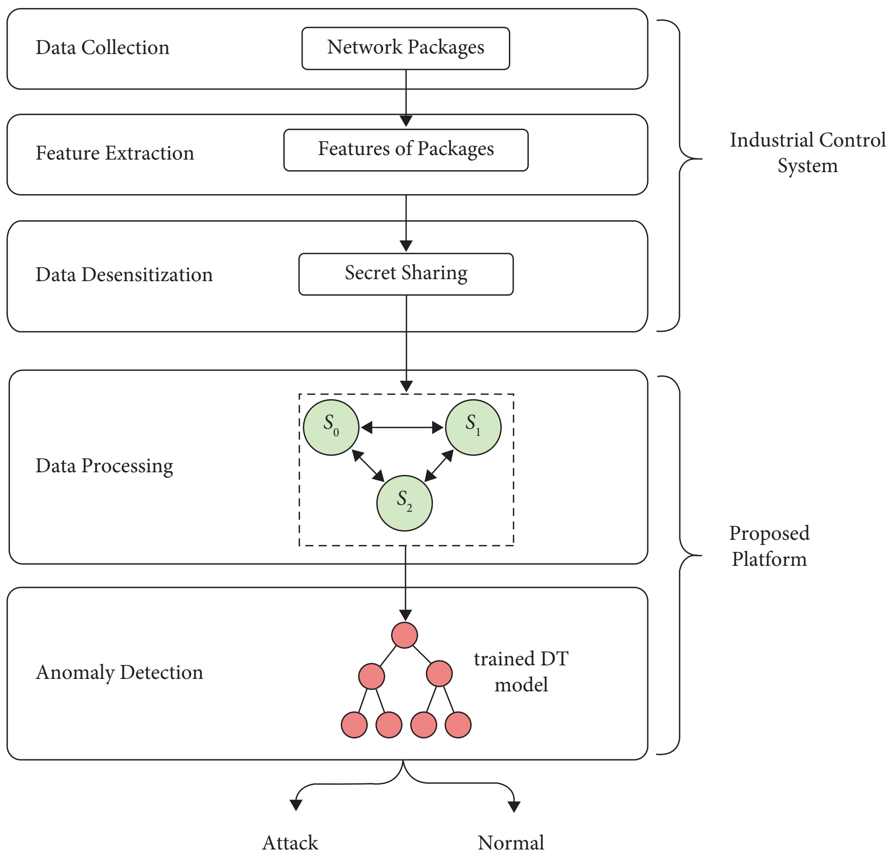

FIGURE 2: System framework.

```
Initialization:
        for each $j \in \mathbb{Z}_3$ do
          $S_j$ and $S_{j+1}$ have the same random seed $\varphi_j \longleftarrow \{0,1\}^{\tau}$;
        end
Preparing:
        for each $S_j$ do
          Generate $\mu_j \longleftarrow \mathbb{Z}_n$;
          Generate a pair of keys $((\mathscr{F}_{\mu_j})_0, (\mathscr{F}_{\mu_j})_1)$ for $f_{\mu_j,1}: \mathbb{Z}_n \longrightarrow \mathbb{Z}_{2^{\ell}}$;
          Send (sid, $(\mathscr{F}_{\mu_j})_0$) to $S_{j+1}$, (sid, $(\mathscr{F}_{\mu_j})_1$) to $S_{j+2}$;
        end
for each $S_j$ do
   Receive (sid, $(x)_j$, $(x)_{j+1}$, $(i)_j$) from the environment;
end
for each $S_j$ do
   for each $k \in \mathbb{Z}_3$ do
      $r_{k,j} \longleftarrow \mathrm{PRF}^{\mathbb{Z}_n}_{\varphi_j}(\mathrm{sid}, k), r_{k,j+2} \longleftarrow \mathrm{PRF}^{\mathbb{Z}_n}_{\varphi_{j+2}}(\mathrm{sid}, k)$;
      $(\sigma_k)_j \longleftarrow (i)_j + r_{k,j} - r_{k,j+2} \pmod n$;
   end
   $(\sigma_j)_j \longleftarrow (\sigma_j)_j - \mu_j \pmod n$;
   Send (sid, $(\sigma_j)_j$, $(\sigma_{j+1})_j$) to $S_{j+2}$, (sid, $(\sigma_j)_j$, $(\sigma_{j+2})_j$) to $S_{j+1}$;
end
for each $S_j$ do
   Receive (sid, $(\sigma_{j+1})_{j+1}$, $((\sigma_{j+2})_{j+1}$) from $S_{j+1}$, (sid, $(\sigma_{j+2})_{j+2}$, $((\sigma_{j+1})_{j+2}$) from $S_{j+2}$;
end
for each $S_j$ do
   for each $k \in \mathbb{Z}_3$ do
```

PROTOCOL 1: Continued.

```
        $\sigma_k \longleftarrow (\sigma_k)_0 + (\sigma_k)_1 + (\sigma_k)_2$;
    **end**
    $\left\{(\beta_{0,\mu_{j+1}})_1, (\beta_{1,\mu_{j+1}})_1, \ldots, (\beta_{n-1,\mu_{j+1}})_1\right\} \longleftarrow$ DPF.EvalAll$(1, (\mathscr{F}_{\mu_{j+1}})_1)$;
    $\left\{(\beta_{0,\mu_{j+2}})_0, (\beta_{1,\mu_{j+2}})_0, \ldots, (\beta_{n-1,\mu_{j+2}})_0\right\} \longleftarrow$ DPF.EvalAll$(0, (\mathscr{F}_{\mu_{j+2}})_0)$;
    $(y)_j \longleftarrow \sum_{k=0}^{n-1}((x_{k+\sigma_{j+1}})_j \cdot (\beta_{k,\mu_{j+1}})_1 + (x_{k+\sigma_{j+2}})_{j+1} \cdot ((\beta_{k,\mu_{j+2}})_0)$;
    $\psi_j \longleftarrow \mathrm{PRF}_{\varphi_j}^{\mathbb{Z}_{2^\ell}}(\mathrm{sid})$, $\psi_{j+2} \longleftarrow \mathrm{PRF}_{\varphi_{j+2}}^{\mathbb{Z}_{2^\ell}}(\mathrm{sid})$;
    Return $(y)_j := (y)_j + \psi_j - \psi_{j+2} \pmod{2^\ell}$;
**end**
```

Protocol 1: 3-party shared OT protocol.

depicted in Figure 3 when storing the tree model. Then, we get a full binary tree such that adding tree nodes does not affect the final result. This full binary tree can be saved as two vectors $\mathbf{N} = (N_0, N_1, \ldots, N_{\mathscr{N}-1})$ and $\mathbf{L} = (l_0, l_1, \ldots, l_{\mathscr{L}-1})$, where $\mathscr{N} := 2^{d-1} - 1$ is the number of non-leaf nodes and $\mathscr{L} := 2^{d-1}$ is the number of leaf nodes in a full binary tree with depth d. N_i $(N_i := \{t_i, v_i\})$ denotes the non-leaf node with index i. The index increases from top to bottom, left to right. If a non-leaf node has non-leaf sub-nodes, its left child node is N_{2i+1} and its right child node is N_{2i+2}. The values t_i and v_i belongs to the i-th non-leaf node. Given a feature vector, the decision tree algorithm extracts the t_i-th value of the feature vector to compare with the threshold v_i. If t_i-th value of feature vector is greater than v_i, perform the same operation on the right child node, otherwise on left child node. The algorithm will be end when the node is a leaf node. The l_i in $\mathbf{L}$ is classification result when the leaf node with index i is the end of decision path. $\mathbf{N}$ and $\mathbf{L}$ are shared as $[\mathbf{N}] = \{(\mathbf{N})_0, (\mathbf{N})_1\}$ and $[\mathbf{L}] = \{(\mathbf{L})_0, (\mathbf{L})_1\}$. Then, S_0 holds $\{(\mathbf{N})_0, (\mathbf{L})_0\}$ and S_1 holds $\{(\mathbf{N})_1, (\mathbf{L})_1\}$, where $(N)_j = (\{(t_0)_j, (v_0)_j\}, \{(t_1)_j, (v_1)_j\}, \ldots, \{(t_{\mathscr{N}-1})_j, (v_{\mathscr{N}-1})_j\})$, $(L)_j = ((l_0)_j, (l_1)_j, \ldots, (l_{\mathscr{L}-1})_j)$ and $j \in \mathbb{Z}_2$.

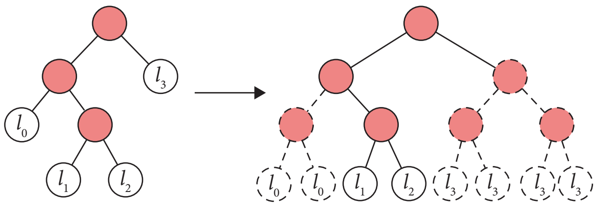

Figure 3: Padding binary tree.

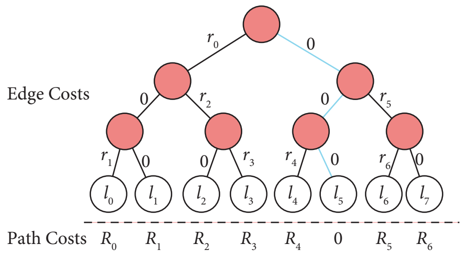

Figure 4: Each edge is set to 0 or random value r_i $(i = 0, 1, 2, \ldots)$. Each path to a leaf node has a cost R_j $(j = 0, 1, 2, \ldots)$ by summing up all costs of edges in the path, and only one's cost is 0.

4.4. Evaluation. When the three servers received a feature vector, respective feature values will be compared with each value v_i in non-leaf node N_i. For each edge in decision tree, S_0 and S_1 will obliviously set their cost to 0 if the edge is selected according to the comparison; otherwise, set to a random non-zero value, as depicted in Figure 4. Then, S_0 and S_1 jointly sum up edge costs for all paths. Among all costs of paths, only one is zero, that is, the corresponding path is the decision process and the classification of the leaf node in this path is detection result.

As described in Protocol 2, the process of evaluation contains three key steps: feature selection, comparison, and path evaluation.

4.4.1. Feature Selection. For each node $N_i = \{t_i, v_i\}$, t_i is stored in S_0 and S_1 in the shared form $[t_i]$. $[t_i]$ will be extended to $\langle t_i \rangle = \{(t_i)_0, (t_i)_1, 0\}$, and S_2 holds 0. Then, run 3-party shared OT mentioned above to get the feature value $\langle x_{t_i} \rangle$.

4.4.2. Comparison. Comparison depends on the DICF scheme, where S_2 generates keys and S_0, S_1 are evaluators. S_2 generates a pair of keys for each non-leaf node $N_i = \{t_i, v_i\}$ to compare corresponding feature value of input with a random value μ_i, such that servers cannot obtain v_i. Then, S_0, S_1 get a value $\Delta x_i = x_{t_i} - v_i + \mu_i$ by jointly computing. Next, S_0 and S_1 jointly get comparison result $[b_i]$ by evaluating DICF keys with Δx_i, as $b_i = 0$ if $(x_{t_i} - v_i) \leq 0$ and $b_i = 1$ otherwise.

4.4.3. Path Evaluation. S_0 and S_1 generate random value r_i together for each non-leaf node N_i in the tree. Then, S_0 and S_1 locally compute the left out-going edge cost $[e_{i,\mathrm{left}}] = [b_i \cdot r_i]$ and the right-going edge cost $[e_{i,\mathrm{right}}] = [(1 - b_i) \cdot r_i]$ for node N_i. Then, as depicted in Figure 4, S_0 and S_1 jointly get path costs $[\mathbf{P}]$, where $\mathbf{P} := (p_0, p_1, \ldots, p_{\mathscr{L}-1})$ and only one path cost in $\mathbf{P}$ is 0. To obliviously get classification result according to the position of 0 in $\mathbf{P}$, S_0, S_1 jointly pick a random value $\delta \longleftarrow \mathbb{Z}_{\mathscr{L}}$, and cyclic shift $\mathbf{L}$ and $\mathbf{P}$ to the

```
Initialization:
    for each $j \in \mathbb{Z}_3$ do
        $S_j$ and $S_{j+1}$ agree on the same random seed $\varphi_j \longleftarrow \{0,1\}^{\tau}$;
    end
Preparing:
    $\varepsilon = 2^{\ell-1} - 1$;
    for each $i \in \mathbb{Z}_{\mathcal{N}}$ do
        $S_2$ generates $\mu_i \longleftarrow \mathbb{Z}_{2^\ell}$;
        $S_2$ generates keys $((\mathcal{F}_{\mu_i})_0, (\mathcal{F}_{\mu_i})_1)$ for $\mathbf{F}_{0,\varepsilon,\mu_i,0}\colon \mathbb{Z}_{2^\ell} \longrightarrow \mathbb{Z}_{2^\tau}$;
        $S_2$ sends (sid, $(\mathcal{F}_{\mu_i})_0$) to $S_0$, (sid, $(\mathcal{F}_{\mu_i})_1$) to $S_1$;
    end
for each $S_j$ do
    for each $i \in \mathbb{Z}_{\mathcal{N}}$ do
        $r_{i,j} \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{2^\ell}}_{\varphi_j}(\mathrm{sid}, i), r_{i,j+2} \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{2^\ell}}_{\varphi_{j+2}}(\mathrm{sid}, i)$;
        $(t_i)_2 \longleftarrow 0$, $(v_i)_2 \longleftarrow \mu_i$;//feature selection
        Run 3-party shared OT protocol to get $(x_{t_i})_j$;
        $(\Delta x_i)_j \longleftarrow (x_{t_i})_j - (v_i)_j + r_{i,j} - r_{i,j+2} \pmod{2^\ell}$;//comparison
    end
    $(\Delta x)_j\colon = ((\Delta x_0)_j, (\Delta x_1)_j, \ldots, (\Delta x_{\mathcal{N}-1})_j)$;
    Send ((sid, $(\Delta \mathbf{x})_j$) to $S_0, S_1$;
end
for each $j \in \{0,1\}$ do
    $S_j$ receives (sid, $(\Delta \mathbf{x})_{1-j}$) from $S_{1-j}$, (sid, $(\Delta \mathbf{x})_2$) from $S_2$;
    for each $i \in \mathbb{Z}_{\mathcal{N}}$ do
        $\Delta x_i \longleftarrow (\Delta x_i)_0 + (\Delta x_i)_1 + (\Delta x_i)_2 \pmod{2^\ell}$;
        $(b_i)_j \longleftarrow \mathrm{DICF.Eval}_{0,\varepsilon}(j, (\mathcal{F}_{\mu_i})_j, \Delta x_i)$;
        $r_i \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{2^\tau}}_{\varphi_0}(\mathrm{sid}, i)$;//path evaluation
        $(e_{i,\mathrm{right}})_j \longleftarrow (1 - j - (b_i)_j) \cdot r_i$, $(e_{i,\mathrm{left}})_j \longleftarrow (b_i)_j \cdot r_i$;
    end
    $\delta \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{\mathcal{L}}}_{\varphi_0}(\mathrm{sid}, 0)$;
    for $i \in \mathbb{Z}_{\mathcal{L}}$ do
        $(p_i)_j \longleftarrow$ Sum up the share of edge costs along $i$-th leaf node's path;
        $(p_i')_j \longleftarrow (p_{i-\delta \pmod{\mathcal{L}}})_j$;
        $(q_i)_0 \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{2^\ell}}_{\varphi_0}(\mathrm{sid}, i, 0)$, $(q_i)_1 \longleftarrow \mathrm{PRF}^{\mathbb{Z}_{2^\ell}}_{\varphi_0}(\mathrm{sid}, i, 1)$;
        $(l_i'')_j \longleftarrow (l_{i-\delta \pmod{\mathcal{L}}})_j - (q_i)_j \pmod{2^\ell}$;
    end
        $(\mathbf{P}')_j\colon = ((p_0')_j, (p_1')_j, \ldots, (p_{\mathcal{L}-1}')_j)$, $((\mathbf{L}'')_j\colon = (l_0'')_j, (l_1'')_j, \ldots, (l_{\mathcal{L}-1}'')_j)$;
        Send (sid, $(\mathbf{P}')_j$, $(\mathbf{L}'')_j$) to $S_2$
end
$S_2$ receives (sid, $(\mathbf{P}')_0$, $(\mathbf{L}'')_j$) from $S_0$, (sid, $(\mathbf{P}')_1$, $(\mathbf{L}'')_1$) from $S_1$;
for each $i \in \mathbb{Z}_{\mathcal{L}}$ do
    if $(p_i)_0 + (p_i)_1 = 0 \pmod{2^\tau}$ then
        $\omega \longleftarrow i$;
        $((\mathcal{F}_\omega)_0, (\mathcal{F}_\omega)_1 \longleftarrow \mathrm{DPF.Gen}(1^\tau, f_{\omega,1}))$ for point function $f_{\omega,1}\colon \mathbb{Z}_{\mathcal{L}} \longrightarrow \mathbb{Z}_{2^\ell}$;
        Send (sid, $(\mathcal{F}_\omega)_0$) to $S_0$, (sid, $(\mathcal{F}_\omega)_1$) to $S_1$;
        Return $l_\omega''\colon = (l_\omega'')_1 + (l_\omega'')_2 \pmod{2^\ell}$;
    end
end
for each $j \in \{0,1\}$ do
    $S_j$ receives (sid, $(\mathcal{F}_\omega)_j$);
    $\{(\beta_0)_j, (\beta_1)_j, \ldots, (\beta_{\mathcal{L}-1})_j\} \longleftarrow \mathrm{DPF.EvalAll}(j, (\mathcal{F}_\omega)_j)$;
    $(q_\omega)_j \longleftarrow \sum_{i=0}^{\mathcal{L}-1}(((q_i)_0 + (q_i)_1) \cdot (\beta_i)_j) \pmod{2^\ell}$;
    Return $(q_\omega)_j$;
end
```

Protocol 2: Constant-round evaluation protocol.

right δ position to obtain $\mathbf{L}' = (l_0', l_1', \ldots, l_{\mathcal{L}-1}')$ and $\mathbf{P}' = (p_0', p_1', \ldots, p_{\mathcal{L}-1}')$. Then, they generate a random vector $\mathbf{Q}\colon = (q_0, q_1, \ldots, q_{\mathcal{L}-1}) \longleftarrow (\mathbb{Z}_{2^\ell})^{\mathcal{L}}$, and jointly compute $\mathbf{L}'' = \mathbf{L}' - \mathbf{Q}$ ($l_i''\colon = l_i' - q_i$ is the element with index i in $\mathbf{L}''$). Subsequently, S_0, S_1 reveal $\mathbf{P}', \mathbf{L}''$ to S_2. After obtaining $\mathbf{P}', \mathbf{L}''$, S_2 generates a pair of DPF keys for point function $f_{\omega,1}(x)$, where ω is the index of $p_\omega' = 0$. Lastly, S_0, S_1 serve as evaluators and jointly compute $[q_\omega] = \sum_{i=0}^{\mathcal{L}-1}(q_i \cdot$

$[f_{\omega,1}(i)])\ (\mathrm{mod}\ 2^{\ell})$. S_0, S_1 send $[q_\omega]$ to ICS and S_2 sends l''_ω to ICS, so that the system admin of ICS gets classification result. Note that the result $l'_\omega = q_\omega + l''_\omega\ (\mathrm{mod}\ 2^{\ell})$.

4.5. Security Analysis. The main building block of our privacy-preserving decision tree evaluation protocol is the 3-party shared OT (cf. Section 4.1). The construction of the 3-party shared OT protocol is inspired by Paul et al. [14]. At high level, in turn, each of the three servers plays the role of a DPF generator, and the other two servers play the role of DPF evaluators to obtain the i-th position value of their (2, 3)-replicated shared data $\mathbf{x}$. In particular, for instance, when S_0 is the DPF generator, it generates a pair of DPF keys for a random position $\mu \in \mathbb{Z}_n$. Let the shared OT choice be $[i]$. In the online phase, the servers jointly open $\delta := i - \mu\ (\mathrm{mod}\ n)$ to S_1, S_2. They can then shift the shared data $\mathbf{x}$ by δ position such that the μ-th position of the shifted data $\mathbf{x}'$ is x_i.

We now analyze the security of this part. First of all, revealing $\delta := i - \mu\ (\mathrm{mod}\ n)$ leaks no information about i, as i is masked by μ information theoretically. Moreover, assuming that the underlying DPF scheme is secure, S_1 and S_2 obtain the shared form of x_i. Repeating the above process for all three servers, we obtain the final result. Note that, for efficiency, we use PRF to generate shares of 0 without communication; assume that the underlying PRF is secure, and the generated shares are computationally indistinguishable from uniformly random ones. Therefore, when any of the three servers is semi-honest corrupted, its view is computationally indistinguishable from a few random shares (and the DPF keys).

When the model is not a full binary tree, we pad the model to a full binary tree by adding dummy nodes; therefore, the tree evaluation process does not leak any information about the tree structure to the MPC players. The security of the feature selection phase can be reduced to the security of the 3-party shared OT. In addition, we adopt the DICF scheme for secure comparison, and its security is proven in [11]. Finally, with regard to the path evaluation, we designed an encoding scheme for the tree such that we can evaluate the path within one multiplicative round. As a result, after evaluation, only the output label will be 0, and the remaining labels are uniformly random. Since each tree uses different fresh random encoding instances, the three severs cannot learn any additional information other than the intended output label.

5. Implementation and Benchmark

5.1. Dataset Description and Experiment Setup. In the experiment, we adopt CICIDS2017 [15] as dataset. It captures normal packages and attacks in simulated network environment that is similar to the real-world network. The dataset contains several CSV files, and each of them includes a kind of attack. We perform experiment on the CSV files corresponding to DDoS, DoS, botnet, port scan, and web attacks, as described in Table 2. Each entry in CSV file contains a feature vector and a class label. The feature vector includes 78 features, such as timestamp, source IP, destination IP, package length, and protocol.

Table 2: Description of CSV files of CICIDS2017.

Attack type	Normal samples	Attack samples	Total samples
DoS	440031	252661	692692
DDoS	97718	128027	225745
Botnet	189067	1966	191033
Port scan	127537	158930	286467
Web attacks	168186	2180	170366
Total	1022539	543764	1566303

We evaluate the performance of the GBDT model and CART model for the binary classification task with CICIDS2017 dataset. The samples of DDos, DoS, botnet, port scan, and web attacks are labeled as attack, and the others are labeled as normal. We randomly select 80% of this dataset as our training set and the remainder as validation set.

5.2. Evaluation Metrics. To evaluate the performance of intrusion detection, we adopt some evaluation metrics, such as Precision, Recall, and F_1. These metrics depend on four parameters. (i) True Positive (TP) denotes the number of attack samples that are correctly classified. (ii) False Negative (FN) denotes the number of attack samples that are wrongly classified. (iii) True Negative (TN) denotes the number of normal samples that are correctly classified. (iv) False Positive (FP) denotes the number of normal samples that are wrongly classified.

(i) Precision indicates how many samples that are classified as attacks are real attacks.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{8}$$

(ii) Recall indicates how many attack samples are correctly classified. Since the proportion of attack in the total sample is small and the attack will cause severe consequence, we need to identify as many attacks as possible. Therefore, Recall is an important evaluation metric.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{9}$$

(iii) F_1-score is calculated based on Precision and Recall and shows the trade-off between Precision and Recall.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{10}$$

5.3. Intrusion Detection Result. In our experiment, we utilize GBDT mentioned in Section 2.2 as our detection model firstly. We set the regularization coefficients $\gamma = 1$ and $\lambda = 1$ and learning rate as 0.1. In the GBDT model, each tree's maximum depth is set to 9. Figure 5 shows the performance of GBDT model, when the iterations are 5, 10, 15, 20, 40, and 60.

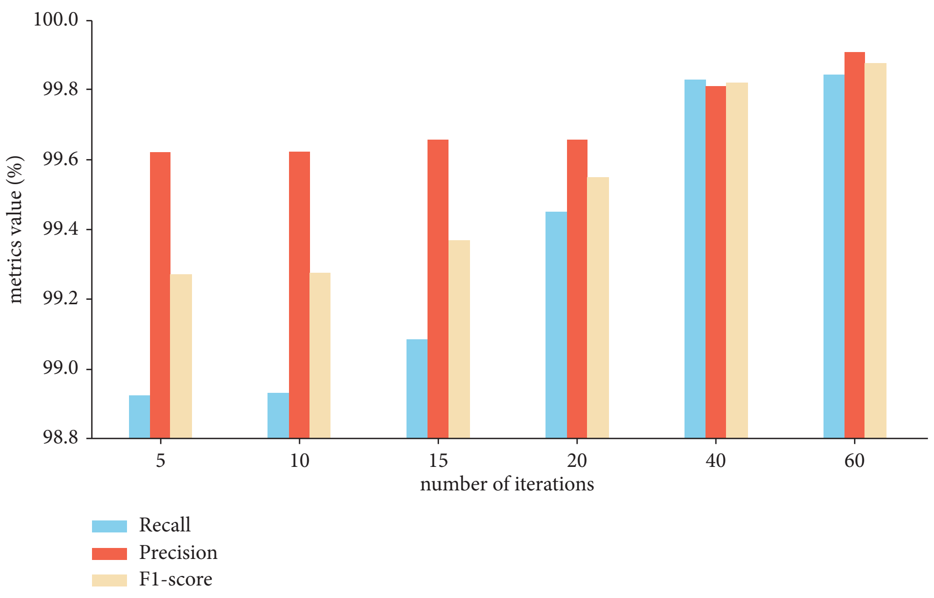

Figure 5: The anomaly detection result using GBDT model with different number of iterations to detect samples in validation set.

We use regularization item to prevent overfitting. As shown in Figure 5, the metrics Recall and F_1-score notably enhance as the number of iterations increases. When the number of iterations reaches 60, the GBDT model performs good on three metrics (99.84% Recall, 99.90% Precision, and 99.87% F_1).

Then, we use a CART decision tree model as detection model to evaluate its performance on CICIDS2017 dataset. We set the maximum depth of decision tree to 9 and obtain its evaluation result (99.09% Recall, 94.32% Precision, and 96.65% F_1). By adopting multiple decision trees for iterative learning, the GBDT model obtain stronger classification ability than single CART decision tree.

5.4. Time Efficiency. We run our platform in different network environments to evaluate its time efficiency. The network environments are simulated, including LAN (0.1 ms RTT, 1 Gbps bandwidth), MAN (6 ms RTT, 100 Mbps bandwidth), and WAN (80 ms RTT, 40 Mbps bandwidth). We set the depth of full decision tree to 5, 7, 9, 11, and 13. We evaluate the time efficiency of our proposed platform when the platform evaluates one tree and evaluates one thousand trees. Each tree is a full binary tree. Our benchmarks are executed on a desktop with Intel(R) Core i7 8700 CPU @ 3.2 GHz, and the operating system is Ubuntu 18.04.2 LTS with 6 CPUs, 32 GB memory, and 1 TB SSD.

As shown in Table 3, our platform performs good in different simulated network environments. Our platform can evaluate one thousand trees whose depth is 9 in 11 seconds when the network environment is LAN (0.1 ms RTT, 1 Gbps bandwidth). However, the increasing depth of decision tree results in more communication cost because the constant-round protocol's communication cost is $O(2^d)$, where d is depth of full tree. Therefore, the proposed protocol is not suitable for the tree model whose depth is greater than 9. As GBDT uses multiple trees, the GBDT model is significantly slower than the CART model. Each tree of GBDT can be evaluated independently, and finally client sums up trees' evaluation result to obtain classification result. Therefore, we can improve the parallel computing capability of the proposed platform to enhance time efficiency of the GBDT model.

6. Related Work

Anomaly detection has been developed for decades and is widely used as defensive method in conventional network. However, since ICS is different from conventional network system, anomaly detection technique cannot be used in ICS directly. Availability and real-time performance are required in ICS-specific IDS [16]. There are a large number of works on ICS-specific IDS. With the development of machine learning (ML) and deep learning (DL) algorithms, most recent works use them to detect anomaly in ICS. The authors in [17] evaluated several machine learning models on an ICS dataset called Power System Dataset, such as Nearest Neighbor, Random Forests, Naive Bayes, SVM, AdaBoost, and JRip. In [18], the authors evaluated different ML and DL algorithms using their generated ICS dataset Electra. These algorithms contain One-Class SVM, SVM, Isolation Forest, Random Forest, and Neural Network. In [19], the authors used the Pearson Correlation Coefficient (PCC) to select packet features and used the Gaussian Mixture Model (GMM) to transform important features for privacy preservation. Then, they used the transformed features as input of a Kalman Filter to detect anomaly. In [20], they utilized Bloom filter to store the signature database for packet-based intrusion detection and applied an LSTM model to learn temporal features.

In private branching program (BP) and decision tree evaluation with constant communication round, there have been several works. The work in [21] evaluates BP with input encrypted by homomorphic public-key cryptosystem.

Table 3: Time efficiency (ms) of the proposed platform when setting different tree depths in different network environments.

Maximum depth		5	7	9	11	13
Evaluating 1 tree	1000mbps/0.1ms	5.1	6.3	9.3	28.1	28.8
	100mbps/6ms	50.6	52.2	55.6	71.4	72.2
	40mbps/80ms	408.9	409.2	410.0	425.7	476.1
Evaluating 1000 trees	1000mbps/0.1ms	4226.9	6713.2	11002.4	29790.6	31540.4
	100mbps/6ms	51990.1	52644.1	54541.3	73736.8	75969.1
	40mbps/80ms	385858.2	391692.5	406701.5	413470.7	413940.0

The maximum depth of tree is set to 5, 7, 9, 11, and 13.

However, it is impractical when the input feature vector is too large. After that, some evaluation protocols with constant communication round are proposed. In [22], the authors utilized additive homomorphic encryption (AHE) and OT for obliviously feature selection and converted the BP model into a secure program with Garble Circuits for comparison. Bost et al. [23] evaluated a decision tree with costly fully homomorphic encryption (FHE) by treating decision tree as a high-degree polynomial. The authors of [24] used OT to select leaf node and DGK protocol based on AHE instead of FHE for comparison. Raymond et al. [25] improved the work in [24] by representing decision tree as linear functions instead of high-degree polynomial form. They computed "path cost" of each leaf node and used it to decide which leaf node contains classification result. In [26], they reviewed prior constant-round approaches and proposed a modular construction from three constant-round sub-protocols: private feature selection, secure comparison, and oblivious path evaluation.

7. Conclusion and Future Work

In this paper, we proposed a privacy-preserving anomaly detection platform for industrial control system. It depends on two main components, detection model and MPC protocol. We use GBDT and CART as anomaly detection models, which are able to detect anomaly with high accuracy. As information privacy is protected by laws and regulations in many countries, we adopt a MPC protocol that can detect network packages from ICS based on decision tree when sensitive data are invisible. The experimental results indicate that the proposed platform can detect anomaly on package level in real time with high accuracy.

Our platform can be developed in several ways in the future. Firstly, we plan to evaluate the performance of our platform in a simulated environment that resembles real environment. In addition, to make detection model more practical, it is necessary to use real data of ICS as training set. Lastly, we will utilize a privacy-preserving machine learning approach in training stage to ensure training data privacy.

Data Availability

The experiment data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Key R&D Program of China (no. 2021YFB3101601), the National Natural Science Foundation of China (grant no. 62072401), and the Open Project Program of Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province. This project was also supported by Input Output (iohk.io).

References

[1] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *Proceedings of the IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 4490–4494, IEEE, Melbourne, Australia, November, 2011.

[2] A. Hobbs, *The Colonial Pipeline Hack: Exposing Vulnerabilities in Us Cybersecurity*, SAGE Publications, Thousand Oaks, CA, USA, 2021.

[3] P. Voigt and A. Bussche, "Practical implementation of the requirements under the gdpr," in *The EU General Data Protection Regulation (GDPR)*, pp. 245–249, Springer, Berlin, Germany, 2017.

[4] E. Goldman, *An Introduction to the california Consumer Privacy Act (Ccpa)*, Santa Clara Univ. Legal Studies Research Paper, Santa Clara, CA, USA, 2020.

[5] K. Xu, "The effectiveness and function of the personal information security specification," *China Information Security*, vol. 4, no. 3, 2019.

[6] H. Gao, L. Yuan, F. Yin, and G. Shen, "Epcad: efficient and privacy-preserving data anomaly detection scheme for industrial control system networks," in *Journal of Physics: Conference Series*vol. 1856, IOP Publishing, Article ID 012028, 2021.

[7] A. C.-C. Yao, "How to generate and exchange secrets," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pp. 162–167, IEEE, Toronto, Canada, October, 1986.

[8] Y. Huang, D. Evans, and J. Katz, *Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?* NDSS, Manhattan, NY, USA, 2012.

[9] P. Mohassel and Y. Zhang, "Secureml: a system for scalable privacy-preserving machine learning," in *Proceedings of the 2017 IEEE symposium on security and privacy (SP)*, pp. 19–38, IEEE, San Jose, CA, USA, May, 2017.

[10] E. Boyle, N. Gilboa, and Y. Ishai, “Function secret sharing,” in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 337–367, Springer, Berlin, Heidelberg, April, 2015.

[11] E. Boyle, N. Chandran, N. Gilboa et al., “Function secret sharing for mixed-mode and fixed-point secure computation,” in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 871–900, Springer, Cham, Switzerland, June, 2021.

[12] M. O. Rabin, *How to Exchange Secrets with Oblivious Transfer*, Cryptology ePrint Archive, New York, NY, USA, 2005.

[13] R. G. Bace and P. Mell, “Intrusion detection systems,” 2001, http://csrc.nist.gov/publications/nistpubs/800-31/sp800-3%201.pdf.

[14] B. Paul, J. Katz, E. Kushilevitz, and R. Ostrovsky, “Efficient 3-party distributed oram,” in *Proceedings of the International Conference on Security and Cryptography for Networks*, pp. 215–232, Springer, Cham, Switzerland, September, 2020.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *ICISSp*, vol. 1, pp. 108–116, 2018.

[16] B. Zhu and S. Sastry, “Scada-specific intrusion detection/prevention systems: a survey and taxonomy,” *Proceedings of the 1st workshop on secure control systems (SCS)*, vol. 11, no. 7, 2010.

[17] S. Pan, T. Morris, and U. Adhikari, “Developing a hybrid intrusion detection system using data mining for power systems,” *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 3104–3113, 2015.

[18] A. L. Perales Gomez, L. Fernandez Maimo, A. Huertas Celdran et al., “On the generation of anomaly detection datasets in industrial control systems,” *IEEE Access*, vol. 7, pp. 177460–177473, 2019.

[19] M. Keshk, E. Sitnikova, N. Moustafa, J. Hu, and I. Khalil, “An integrated framework for privacy-preserving based anomaly detection for cyber-physical systems,” *IEEE Transactions on Sustainable Computing*, vol. 6, no. 1, pp. 66–79, 2021.

[20] F. Cheng, T. Li, and D. Chana, “Multi-level anomaly detection in industrial control systems via package signatures and lstm networks,” in *Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 261–272, IEEE, Denver, CO, USA, June, 2017.

[21] Y. Ishai and A. Paskin, “Evaluating branching programs on encrypted data,” in *Proceedings of the Theory of Cryptography Conference*, pp. 575–594, Springer, Berlin, Heidelberg, August, 2007.

[22] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, “Privacy-preserving remote diagnostics,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 498–507, Alexandria, VI, USA, October, 2007.

[23] R. Bost, R. Ada Popa, S. Tu, and S. Goldwasser, *Machine Learning Classification over Encrypted Data*, Cryptology ePrint Archive, New York, NY, USA, 2014.

[24] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter, “Privately evaluating decision trees and random forests,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 335–355, 2016.

[25] K. H. Raymond, J. P. K. Ma, Y. Zhao, and S. M. Sherman, “Privacy-preserving decision trees evaluation via linear functions,” in *European Symposium on Research in Computer Security*, pp. 494–512, Springer, Berlin, Germany, 2017.

[26] A. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, “Sok: modular and efficient private decision tree evaluation,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 2, pp. 187–208, 2019.