Blockchain-Based AI-Enabled Industry 4.0 CPS Protection Against Advanced Persistent Threat

Ziaur Rahman, Xun Yi, and Ibrahim Khalil

***Abstract*—Industry 4.0 is all about doing things in a concurrent, secure, and fine-grained manner. Internet of Things edge sensors and their associated data play a predominant role in today's industry ecosystem. Breaching data or forging source devices after injecting advanced persistent threats (APTs) damages the industry owners' money and loss of operators' lives. The existing challenges include APT injection attacks targeting vulnerable edge devices, insecure data transportation, trust inconsistencies among stakeholders, incompliant data storing mechanisms, etc. Edge servers often suffer because of their lightweight computation capacity to stamp out unauthorized data or instructions, which, in essence, makes them exposed to attackers. When attackers target edge servers while transporting data using traditional public-key infrastructure-rendered trusts, consortium blockchain (CBC) offers proven techniques to transfer and maintain those sensitive data securely. With the recent improvement of edge machine learning, edge devices can filter malicious data at their end, which largely motivates us to institute a blockchain and artificial intelligence-aligned APT detection system. The unique contributions of this article include efficient APT detection at the edge and transparent recording of the detection history in an immutable blockchain ledger. In line with that, the certificateless data transfer mechanism boosts trust among collaborators and ensures an economical and sustainable mechanism after eliminating existing certificate authority. Finally, the edge-compliant storage technique facilitates efficient predictive maintenance. The respective experimental outcomes reveal that the proposed technique outperforms the other competing systems and models.**

***Index Terms*—Advanced persistent threat (APT), blockchain, deep transfer learning (DTL), edge Internet of Things (IoT), industry 4.0, IoT.**

I. Introduction

SINCE the last decade, the world has experienced the latest iteration of the industrial ecosystem called Industry 4.0. This fourth revolution demands the adoption of connected devices and techniques to meet the increasingly growing system protection requirements. The ultimate goals of building such automated connections range from enhancing productivity, reducing costs to boosting revenue. After merging advanced technology, such as the Internet of Things (IoT), artificial intelligence (AI), etc., the latest industrial infrastructure has laid the foundation for the desired smart factory system, where the convergence happens between machines and humans depending on data. Data generated by edge sensors play a vital role in monitoring the manufacturing process, predicting maintenance, and detecting equipment anomalies. As a critical component, if data fail to comply with the security standard, all the actions associated with data will undoubtedly affect or paralyze the entire industrial ecosystem. The recent security issues published by the Guardian and ABC support the U.S. and Australian claims and concern of stealing their industry copy-right data through cyber espionage by other countries. Even during the worldwide COVID 19 pandemic, Webber has recorded about 50 cyberattacks only in Australia since January 2020, which was 120 in the last two years. The report shows that most attacks targeted large industries, such as Bunnings, Alinta Energy, and Toyota, including sensitive health information. The country indulges a $15 billion package in tackling potential threats and unprecedented loss. U.S. Defense (DoD) also funded 8.5 billion in cybersecurity, with an almost 5% increase over the previous year. On top of the incidents above, security concern has become an inevitable issue that deserves proper addressing inside all processes of the Industry 4.0 system. However, the existing industrial security solutions are mostly designed, relying on the security mechanism in the server side, trust provided by the trusted third party (TTP), such as cloud and certificate service provider.

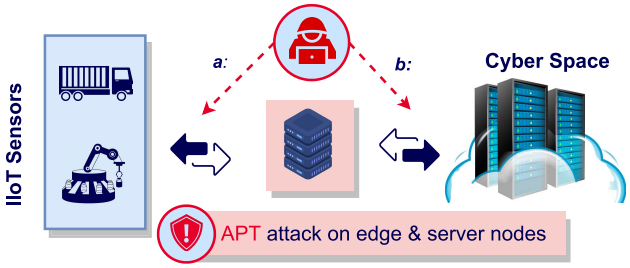

Fig. 1. Cyberattacks to inject APT to the industry 4.0 CPS via (a) IoT edge and (b) cyberspace.

A. APT Attack Model and Challenges

Among several other intrusions and malwares injected to IoT edge nodes, the latest ransomware, namely, advanced persistent threat (APT) has caught broader attention because of

Manuscript received 15 September 2021; revised 16 December 2021; accepted 19 January 2022. Date of publication 28 January 2022; date of current version 7 April 2023. This work was supported by RMIT Research Stipend Scholarship (RRSS) Program. The work of Xun Yi was supported in part by the Project "Privacy-Preserving Online User Matching" under Grant ARC DP180103251. *(Corresponding author: Ziaur Rahman.)*

The authors are with the School of Science, RMIT University, Melbourne, VIC 3001, Australia (e-mail: rahman.ziaur@rmit.edu.au; xun.yi@rmit.edu.au; ibrahim.khalil@rmit.edu.au).

Digital Object Identifier 10.1109/JIOT.2022.3147186

its detective nature. As a stealthy threat actor, an APT strives to control a system network after remaining undetected over a long period. Though attackers target the server side, currently, several incidents were recorded where the edge-side vulnerability was responsible. Therefore, today's Industry 4.0 network has to tackle that it has no APT inside the edge servers. As shown by Fig. 1, APT may enter into the Industry 4.0 cyber–physical system (CPS) both via edge and server nodes. There are several techniques that focus solely server-side protection [1], [2]. Similarly, some works focus edge protection using collaborative machine learning technique. Undeniably, the system cannot be sustainable if there is any security loophole at the edge end [2]. Existing approaches seem to be utilizing complex machine learning algorithms that require significant computational capabilities, which are often NOT edge complaint. Several works have used server-driven data to evaluate their proposed techniques, which may not work at certain circumstances [3].

APT is a well funded, organized group that is systematically developed to compromise large-scale information of government and commercial entities. Malware is any malicious software or program designed to damage or disable computer systems or networks. APT is a broad term used to describe a prolonged, more strategic, and targeted attack. However, most malware attacks are target-specific, quick-damaging attacks. Besides, APT can stay undetected for an extended period; on the contrary, anti-malware tools can detect and eradicate malware from the system.

B. Contributions

With a motivation to protect both edge IoT and server-side data transfer, the key contributions of this article are as follows.

1) A blockchain-based AI-enabled APT detection system is proposed that protect Industrial IoT data from being forged.
2) Reusable machine learning method has been incorporated at the IoT edge that secures data before sending it to the cyberspace.
3) Consortium blockchain (CBC) brings trust among the participating stakeholders that prevent system from centralized dependency and facilitates sustainable system.
4) Certificateless device registration and data transfer technique has been proposed that saves costs after eliminating certificate authority (CA) and brings collaborative operation.
5) Immutable recording of both APT detection and data transaction in the blockchain ledger (BCL) and storing in the edge-complaint distributed hash table (DHT) ensure higher performance and efficiency. Because of the DHT integration, the respective experimental outcomes reveal that the proposed technique outperforms the other system with competing machine learning models.

C. Organization

The remainder of this article is organized as follows. Section II includes the background and related state-of-the-art literature. Section III explains the proposed model where the necessary evaluation is detailed through subsequent Section IV. The final section concludes the future scopes and justifies how authors achieve the claims made throughout the work.

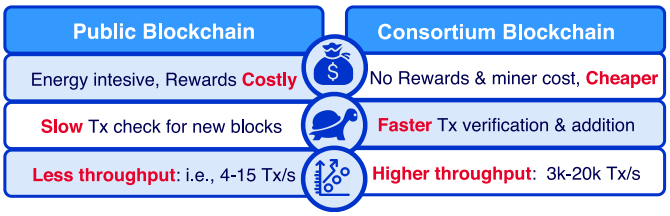

Fig. 2. CBC suitability for industrial IoT.

II. Background and Related Work

Blockchain is a growing, publicly distributed, and permanent ledger to which transaction events are posted and verified by the peers on the network. The entire process happens without the intervein of any third party, which makes it so appealing, indeed. Bitcoin is the most common example of blockchain where data as transactions are maintained after being confirmed through an incentivized system in which members must compete to complete some proof-of-work like cryptographic challenge. One block is linked with its nearest block by using the hash of that block; therefore, any modification in the block breaks all the previous chain and consensus. The latest block establishes the integrity of the last block.

A. Blockchain Suitability for Industry 4.0 CPS

Public blockchain best suits where an utterly untrusted network requires to be safe; however, it is slow and expensive. For example, for setting up a powerful mining node, in reality, is costly, on top of that, it requires enormous energy consumption to process the mining works. Besides, public blockchain can verify only a few transactions per second, which makes it incompatible for the use cases such as industry where plenty of data transactions need to be done in real time. On the other hand, consortium or permissioned type of blockchain, such as Hyperledger (HLF), Quorum, and Corda, has a selective setup where only invited members instead of arbitrary participants are allowed to join the network who agreeably trust each other. Here, token for incentives/rewards is not mandatory; thus, expenses for mining setup can be avoided to make it adaptable for the real time and critical system such as the Industry 4.0 application. As participating nodes are acquainted beforehand, it brings natural protection against "Sybil attacks." Fig. 2 shows that permissioned blockchain (BC) is cheaper, faster and also has higher transaction processing rate. For example, Proof of Work (PoW)-driven BC, i.e., Ethreum, has the rate of four to five transactions per second (Tx/s) where permissioned BC, i.e., HLF Fabric, can process about 3000–20 000 Tx/s, which, in essence, make it an inevitable choice for the proposed industry 4.0 edge communication [4].

B. Deep Transfer Learning

Deep transfer learning (DTL) converges storing knowledge gained while solving one problem and applying it to another,

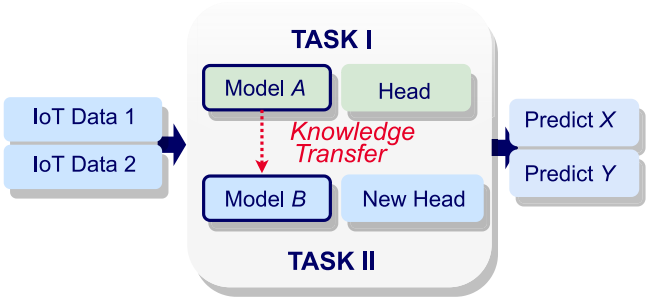

Fig. 3. Knowledge transfer and prediction technique on different data sets in DTL approach.

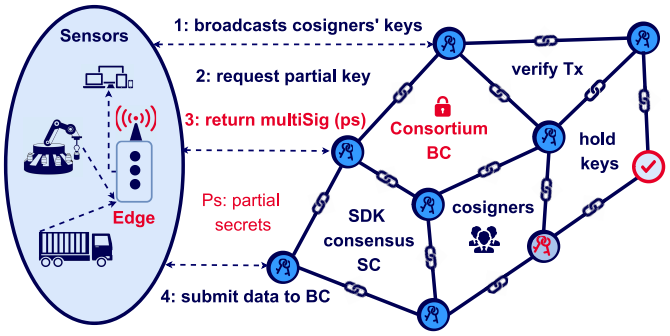

Fig. 4. Certificateless communication between blockchain consortium (KGD) and industry 4.0 IoT edge devices.

i.e., the knowledge to detect malware top up the knowledge of the model that detects intrusion. Fig. 3 depicts how a model transfers/reuses its knowledge to predict a decision in cooperation with another model performing different task. DTL is given in terms of domains and tasks. Suppose a domain $\mathcal{D}$ consists of: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, with $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. Let a domain, $\mathcal{D} = \{X, P(X)\}$, be an example of task with two elements. A label space $\mathcal{Y}$ and objective predictive function $f : \mathcal{X} \rightarrow \mathcal{Y}$. f predicts the respective label $f(x)$ of an instance x. This task, denoted by $\mathcal{T} = \{\mathcal{Y}, f(x)\}$, can be obtained from the training data set consisting of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$ [5], [6].

Assuming an input domain $\mathcal{D}_S$ and a training task $\mathcal{T}_S$, an output domain $\mathcal{D}_T$ as if $\mathcal{D}_S \neq \mathcal{D}_T$ OR $\mathcal{T}_S \neq \mathcal{T}_T$. DTL improves learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$. Besides, how to store edge sensor data deserves illustrations.

C. DHT for Edge Data Storage

Storing data in the DHT and its associated pointer address into BCL best suits for the IoT edge, i.e., smart energy, implantable medical system, car, or any other industry 4.0 CPS, etc., because of its salient features. In our proposed setup, when any external user asks for data access, the key generating and distribution (KGD) authenticates in cooperation with the required number nodes running the CBC network. It confirms the unique benefits, such as traceability, accountability, removing trusted party, decentralized mechanisms, etc., over existing the cloud-driven centralized storage model. The efficient DHT adaption additionally makes the proposed technique robust, self-organizing, highly scalable, and fault tolerant against different attacks, i.e., false query injection attacks, APT, Zeroday, etc. The proposed Industry 4.0 CPS data protection suits most of the DHT protocol, however, the demonstration integrates the interplanetary file system (IPFS) considering the fixed-sized routing, malware, and APT attacks. Before storing data, both transaction and devices need to be authenticated.

D. Certificateless Authentication

Suppose there are three $n = 3$ parties, namely, Bob, Elen, and Peter, in an industry 4.0 setup who agreed to cosign their partial secret *ps* before registering a new device into the system. They are connected over a CBC and work as KGD authority. KGD dissipates their public parameter with the connected Industry 4.0 IoT devices.

The partial key is a concept in ordinary certificateless authenticated encryption (CLAE) and identity-based encryption (IBE) that fixes the key-escrow problem. In the Hypeledger CBC setup, a built-in membership service provider (MSP) works as a CA. The proposed technique replaces it with a public-key infrastructure (PKI)-like key generation center (KGC), by adopting a blockchain (BC) consortium collaborating with the Industry 4.0 CPS peers.

1) Key Generation: Purposing to sign a *ps* for a number of sensor devices, Bob, Elen, and Peter agrees to pick prime numbers p, q, and group generator g (e.g., primitive root) as if $q \mid p$. Private- and public-key pair are the ring elements of $\mathbb{Z}_p$. Let their private keys be x_1, x_2, and x_3, therefore, the calculated public keys will be $y_i = g_i^x$ where $(i = 0, 1, \ldots, n)$ and aggregated public key $Y = \prod_{i=1}^{n} y_i \bmod p$. The key pairs will be $\{p, q, g, Y\}$, $\{x_1, x_2, x_3\}$.

2) Signing Data and Partial Secret: All cosigners choose random number r_i such that $0 < r < \mathbb{Z}_q$ and compute $R_1 = g_i^r$ before finding $R = \prod_i^n R_i \bmod p$. Suppose T is the timestamp, including the dynamic edge identity (E_{id}) formed using all individual sensor ids (ID_j) and other required parameters. Then, the KGD on blockchain will find the signature parameter $c = H(T \| Y \| R \| PS)$, here, $H : M \rightarrow \mathbb{G}_0$ treated as random oracle in the security analysis and *PS* are the KGD generated partial secrets (PSs) for m number of industrial IoT devices at a particular time t that need to be multisigned [4]. Then, the partial signature will be $s_i = (r_i + cx_i) \bmod q$ and the desired multisignature will be (R, S) where $S = \sum_{i=i}^{n} s_i \bmod p$ as demonstrated by Fig. 4.

3) Verifying Device and Data: The device receives the multisignature (R, S) along with the encrypted *ps*. As public-key parameters such as $\{g, Y\}$ besides T are already known to the edge sensors, it produces $c = H(T \| Y \| R \| PS)$ using the same hash algorithm H. The device will accepts *ps* before generating its own public–private key pairs P_j and S_j if and only if it satisfies $g^S \bmod p = R \times Y^c \bmod p$.

E. Related Work

Blockchain immutable nature besides its pseudoanonymity, traceability over the transparent distributed network has

made blockchain an unbeatable tool for Industry 4.0 CPS. Blockchain application is found in the domain of copyright protection of digital data/asset, ID verification/provenance (notarization), real state land ownership transfer, smart-taxation immigration, electronic voting, and privacy-principle compliance (e.g., GDPR [7]). The authors seemed to be practicing the immense benefits of DHT for storing access control and compliance data [8]. However, besides the high-energy conducive miners' incentive disputes, the blockchain network encounters the scalability issues that some existing works concentrated on and aimed at solving through plausible remedies [9]–[11]. Considering the appealing features of blockchain, researchers have incorporated AI (i.e., deep learning, DTL, federated learning, etc.) with it. In an another work, authors propose a cloud-based distributed deep learning framework for phishing and Botnet attack detection and mitigation [12]. A group of authors proposed a permissioned edge blockchain to secure the peer-to-peer (P2P) energy and knowledge sharing in the framework to maximize edge intelligence efficiency [13], where [1] proposes a deep blockchain framework (DBF) designed to offer security-based distributed intrusion detection and privacy-based blockchain with smart contracts in IoT networks.

One of the latest and motivating works proposed a CBC-based framework to protect Industry 4.0 CPS [4]. There are a number of works studied addressed a novel blockchain-enabled model sharing approach to improve the performance of object detection with cross-domain adaptation for automatic driving systems [14]. The authors addressed a special technique, namely, the authentication mechanism based on transfer learning-empowered blockchain, coined ATLB where blockchains are applied to achieve the privacy preservation for industrial applications [2]. Apart from this, a group of researchers proposes a new transfer learning-based secure data fusion (TSDF) strategy for Industry 4.0 like system [15]. Besides focusing the reinforced machine learning scheme [16], another group of authors proposed to enable mobile multiuser to make optimal offloading decisions based on blockchain transaction states and wireless channel qualities. As studied, several mechanisms appear to have limitations to peer with the Industry 4.0 edge IoT protection; however, they have conceptually motivated us to design our proposed technique.

III. Proposed APT Protection Mechanism

The protection scheme proposed here works in three steps. In the first step, a DTL model gets deployed inside the edge server. The model is trained based on two combined and pre-processed data sets [17], [18] and the trained model is settled down in the edge server. Once edge server is called, it checks if there any APTs found within that data. Second, detection history along with the sensor data is sent to the linked DHT. Before storing the data, it needs to check if there any APT injected during the data transfer over the network. In this step, a blockchain consortium administers the process and ensures that only the registered and authenticated devices are sending data. In the final step, blockchain smartcontract records the data transaction into the shared ledger and stores data into the DHT storage. Fig. 5 portrays the steps one by one. The captions briefly shows those, respectively.

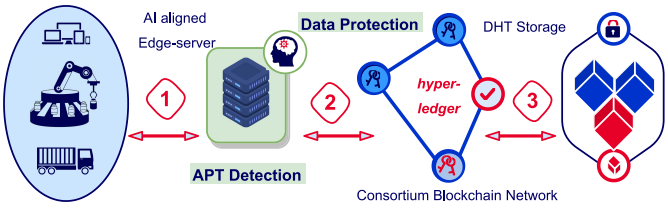

Fig. 5. High-level view of the proposed blockchain-based AI-enabled Industry 4.0 CPS data protection technique. It works within three steps: 1) detecting APT at edge server upon machine learning model; 2) CBC validates none but the authentic sensors sending data; and 3) updating BCL with the data transaction and APT detection status and finally storing data in the DHT storage. Connection establishment mechanism is illustrated in Fig. 4 earlier.

TABLE I
Symbols and Description

Description	Source (s)	Target (t)
Domain data	$D_s : (X_s, Y_s)$	$D_t : (X_t, Y_t)$
Domain feature	X_s	X_t
Domain label	Y_s	Y_t
Number of domain data	n	m

A. Building DTL Model

In order to identify the problems in the IoT environment better, Table I gives the topical symbols and descriptions, in which we set the initial model that has enough labeled data to build an efficient intrusion detection model. When the new complex type of cyberattacks arrives, the detection model is suitable for the new type of cyberattacks [19].

Source Domain: It is the domain where the initial model is located. The source-domain data ($D_s : (X_s, Y_s)$) are the combination of $(X_{s1}, Y_{s1}), (X_{s2}, Y_{s2}), (X_{s3}, Y_{s3}), \ldots, (X_{sn}, Y_{sm})$, in which the class of source-domain label data (Y_s) is 0 and 1, where the normal scenario is represented by 1 and the attack scenario is represented by 0.

Target Domain: The domain has a new type of attacks. The target-domain data ($D_t : (X_t, Y_t)$) is the combination of $(X_{t1}, Y_{t1}), (X_{t2}, Y_{t2}), (X_{t3}, Y_{t3}), \ldots, (X_{tn}, Y_{tm})$, in which the class of target-domain label data (Y_t) is 0 and 1, where the normal scenario is represented by 1 and the attack scenario is represented by 0.

Furthermore, the source-domain label (Y_s) and the target domain label (Y_t) contain only "normal" and "attack" data, but attackers in the source domain and target domain may be different. Although the source-domain label (Y_s) and the target-domain label (Y_t) have the same feature space, their performance in specific features is different. We have used the formula called maximum mean discrepancy (MMD) [20] to measure the difference between the source domain and the target domain

$$\text{Distance}(X_s, X_t) = \left\| \frac{1}{n} \sum_{i=0}^{n} \phi(X_{s_i}) - \frac{1}{m} \sum_{i=0}^{m} \phi(X_{t_i}) \right\|^2.$$

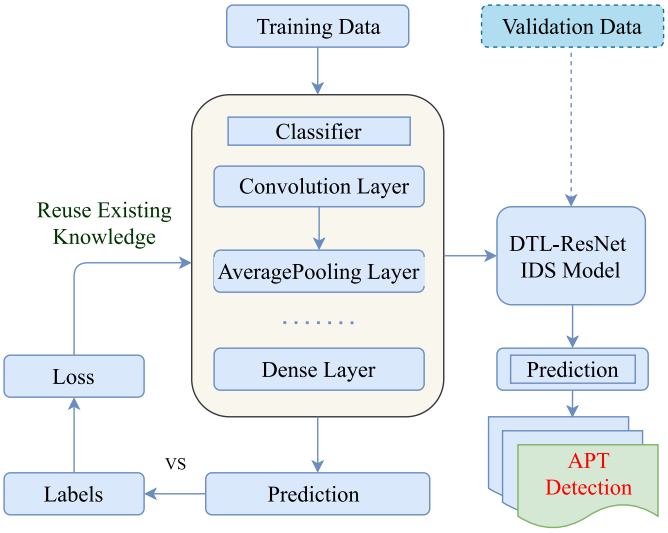

Fig. 6. APT detection flow using DTL ResNet model. The prediction compares with the previous loss and reuse the knowledge. Once model is ready and deployed in the edge server can detect APT injected to the system.

According to the dependence of traditional machine learning (TML) and DL models, the detection model trained by source-domain data (D_s) does not have good detection accuracy when facing target-domain data (D_t), and it has been completely confirmed by the subsequent experiment. The TML and DL models need sufficient training data; thus, it is difficult to train an efficient APT Detection model model only depending on a small scale of target-domain source data (D_t).

Therefore, we transfer the knowledge contained in source-domain data (D_s) to the target domain through the proposed DTL-residual neural network (ResNet) method and combine the target-domain data (D_t) with the same DTL-ResNet method to construct an efficient APT detection for the target domain to improve the detection accuracy for any heterogeneous IoT ecosystems.

B. DTL ResNet APT Detection Technique

Fig. 6 shows the block diagram of the proposed DTL-ResNet-based model for APT detection, which predominantly includes two parts: the first one is the model training part and the last one is the intrusion detection part. We have applied it to our proposed model after prepossessing the network data. The most significant parameters of this model will be determined through subsequent empirical experiments. The model with an optimal prediction performance on the training set will be selected as the final intrusion detection model for heterogeneous IoT applications. As for the intrusion detection part, we have trained the DTL-ResNet model by randomly selected training data set and validated the model by the validation data set. The detection performance of the models under the discrete type of parameters is compared. Finally, the optimal performance model has been selected as the final detection model in the field of heterogeneous IoT applications.

The network architecture that we have selected for the DTL approach is a 1-D fully convolutional neural network (FCN) called *Conv1D*. Fig. 6 shows the architecture of the proposed DTL-ResNet model. The input of the network is the same shape. The 1-D convolution layer is used in the first, second, and third layers, respectively. The first layer is the combination of 128 filters of kernel length 8, the second layer is the combination of 256 filters of kernel length 5, and the third layer is the combination of 128 filters of kernel length 3. The rectified linear-unit (ReLU) activation function is used in the third 1-D convolution layer. Each 1-D convolution layer is followed by a *BatchNormalization*() operation [21]. The combination of these three convolution layers has a stride equal to 1. The procedure repeats two times for block-2 and block 3, respectively, purposing to achieve optimal performance. Each block takes the previous block outputs as inputs for the current block and performs some nonlinearity's to transform it into a multivariate series whose dimensions are defined by the number of filters in each layer. The fourth layer is the combination of a *GlobalAveragePooling1D*() operation, which takes the input of the third block and averages each series. This operation reduces drastically the number of parameters in a deep model while enabling the use of a class activation map [22], which allows an interpretation of the learned features [21]. The output of the gap layer is then fed to a *softmax* classification layer whose number of neurons is equal to the number of classes in the data set. Other hyperparameters and data correlation are skipped purposing brevity of this article. This enabled us to identify the effect of DTL in the field of APT detection. However, once APT detection successfully runs in the edge server, the detection status is kept encrypted purposing to send it during the data transfer over the CBC network. Suppose a detection status is T_s and data transaction is T_x. This will be recorded in the BCL. As mentioned earlier, the CBC works as KGD, which establishes the communication between edge devices and blockchain peers. Though existing CBC, i.e., hyperledger fabric (HLF) [7], works in cooperation with the MSP, which is actually a PKI-driven CA, the proposed technique replaces the need of CA need by facilitating a novel certificateless technique using Elliptic curve-powered multisignature (MS), i.e., BLS/Schnorr variant, etc. The following section explains how it fullfills that requirements.

C. Blockchain-Based Certificateless Authentication

IBE encounters crucial key escrow issues while it emerges with the legacy of PKI. As discussed in the previous section, certificate-less cryptography (CLC) solved it by introducing the PS concept, derived from a master secret (MS) that keeps the private keys apart from blockchain consortium (KGD) as already mentioned in Fig. 4. (*ps*) depends on the device identity, which furthermore ensures the mutual dependency instead of sole reliance on trusted TTP, i.e., CA. In Industry 4.0 use case, once a device receives the partial secret (*ps*), it starts generating public–private key pairs (Pk, Sk) using its identities (*Ids*). As depicted by the four steps (a: to d:) interaction between A and B of Fig. 4, the entire key generation tasks can be divided into the following five consequent processes.

Setup (1^{λ}) $\rightarrow$ (y, ms)*:* It takes a system's security parameter λ and returns the system parameter y and master secret (*ms*).

The algorithm associated with this procedure runs at (KGD). For example, for the customized Schnorr multisignature (MS), y includes the prime numbers(p, q), group generators (g), i.e., primitive roots, etc. It finalizes the after confirming each peer's own private keys such as (x_1, x_2, x_3), master secret (ms), and public keys such as (p, q, g, y) as mentioned in the earlier key generation section.

genPS$(y, id_j, ms) \rightarrow (ps_j)$*:* As illustrated by Algorithm 1, the algorithm takes inputs of system parameter y, j'th number of identities (id_j) that is interested to join at a certain time t, along with the previously created master secret (ms). Here, the device identity $id_j \in \{0, 1\}^*$, and ms outputs the j'th number of partial keys (ps_j) in response. In line with that it takes (id_j) and finds device secret value $x_j \leftarrow (y, id_j)$.

multiSig$(x_i, ps_j) \rightarrow (R, S)$: At this stage, blockchain peers cosign the partial secret (ps_j) before sending it to the IIoT sensors through edge platform. Initially, they compute the shareable parameters $R_i \leftarrow g_i^r$ after choosing own secret random number r_i. Then, it aggregates $R \leftarrow \prod_i^n R_i \text{mod} p$. Besides, it computes $c \rightarrow (T \| Y \| R \| ps_j)$, individual signature, $s_i \leftarrow (r_i + cx_i) \bmod q$, and finally, aggregate $S \leftarrow \sum_{i=i}^n s_i \bmod p$. Therefore, the signature outcomes as if $\sigma \leftarrow (R, S)$.

genSk$(y, ps_j, x_j) \rightarrow SK_j$*:* At the beginning of this step, the edge devices should know the system security parameters Y, which includes the prime numbers (p, q) group generating primitive roots g and calculates public-key variant y. Then, edge gateway on behalf of device or the each IIoT device itself verifies if the partial keys received are valid or not simply by justifying the conditions such as $g^S \bmod p = R \times y^c \bmod p$. If the multisigned partial keys (ps_j) are valid, the particular IIoT device runs its algorithm to produce its own private keys (sk_j).

genPk$(y, x_j) \rightarrow pk_j$*:* To generate the public keys (pk_j) for the IIoT devices, it takes system parameters along with the previously generated device secret x_j. Note that though it looks similar but the public generation for the devices is not the same process as the key generation of the blockchain peers as discussed earlier.

The above procedures should work in line with some base method, such *encrypt*(), *decrypt*(), and *verify*(), along the associated algorithms that could be simplified as follows.

1) *enc*$(y, ps \in m, id, pk) \rightarrow (c \in \mathbb{c} \vee \perp)$*:* The base *encryption* method takes the system parameters y and PSs as message (m) and produces the desired ciphertext (c) within the designated ciphertext space.
2) *dec*$(c \in \mathbb{c}, sk \rightarrow ps \in m \vee \perp)$*:* The base *decryption* method and its associated algorithm take the ciphertext and produces the partial secret (ps) within the designated message space.
3) *ver*$(y, \sigma, id, ps) \rightarrow$ *true(1)* $\vee$ *false(0)* $\vee \perp$*:* The verification algorithm in the edge side,takes the system security parameters (y), the multisigned signature (σ), device identities (id), and the PSs, hence, verifies either the multisigned PSs are valid $(true \vee 1)$ or invalid $(invalid \vee 0)$.

D. Device Registration and Verification

KGD plays an indispensable rule for security of the edge sensors. The sensor devices need to get registered with the KGD consortium. Before posting the transaction (t_x), interested sensors obtain public–private key pairs upon the completion of the registration process.

Algorithm 1: Certificateless Device Key Generation

```
Input  : id_j – identities of the j'th number of IIoT devices
         y – system parameters                    /* prime numbers */
Output: (pk, sk)  – public and private key pairs
setup(1^λ)→ (y)                                   /* sys param init */
for id ← id_j do
    porocedure keyGen (y, id):                    /* key gen */
    X_j ← genSk (y, id_j)                         /* gensec keys */
    requestSend (id_j)                            /* request to join */
    ps_j ← genPS (id_j)                           /* KGDs gen partial sec */
    multiSig(s_n, id_j, ps_j)                     /* multi-sig */
    responseReceived (id_j)                       /* get ps */
    V [0,1,⊥ ] ←verify ()                         /* verify sig */
    if V ← 1 then
        sk_j ← genSk (Y, id_j, x_j)               /* set pri key */
        pk_j ← genPk (y, x_j)                     /* sets pub key */
    end
end
```

1) Registering Devices: As discussed earlier, multiparty industry 4.0 stakeholders agrees to build the KGD cooperatively. Suppose, the owner, buyer, and insurer along with other stakeholders cooperatively form the BC network that represents the KGD peers. They peers broadcasts the system parameters (y) to all sensors. KGD peers keep their individual signer's secret such as $s_1, s_2, \ldots, s_n$ With the help of edge computation capacity or its own ability, interested device creates their own secret value $x_1, x_2, x_3, \ldots, x_j$ that generates respective public keys using x_j and the system parameter y, where j is the number of interested devices at particular time t and n is the number of co-signing blockchain peers. Sensor devices will contact the KGD with their identities $id_1, id_2, \ldots, id_j$. Upon receiving the request, KGD will generate a partial private seccret $ps_1, ps_2, \ldots, ps_i$ for all requested devices and will cosign co-sign id_i and ps_i using co-signers private key s_n. KGD sends the signed message back to the edge. The sensor device itself or edge node (e.g., Azure IoT edge or Dell Gateway) will verify if the message comes from the KGD, and if yes, it will generate private–private key pairs $(pk_1, pk_2, \ldots, pk_i, sk_1, sk_2, \ldots, sk_i)$ using (ps_i, x_i, Y). Note that only each industrial IoT device will be able to create the private key because it is the only entity who knows his private secrets x_j. Algorithm 1 illustrates the step by process with the necessary explanations.

E. Data Protection and Storing

Once edge sensors are successfully registered to the KGD upon the CLC and multisignature-based authentication, the sensor devices proceed further to send and store data as illustrated by Fig. 7. The transaction includes the identity of the IIoT devices along with the action and timestamp at the time (T) of action (ACT). There can be different types of actions such as *store* data at a specific DHT address (ADS), *update* previously inserted data, or *access* permission of the particular data. To verify a transaction $T_x = (\text{ID}_j, T, ACT)$, the blockchain peers have to meet two conditions: 1) either the public key (PK_j) obtained associates with the identity (ID_j) and

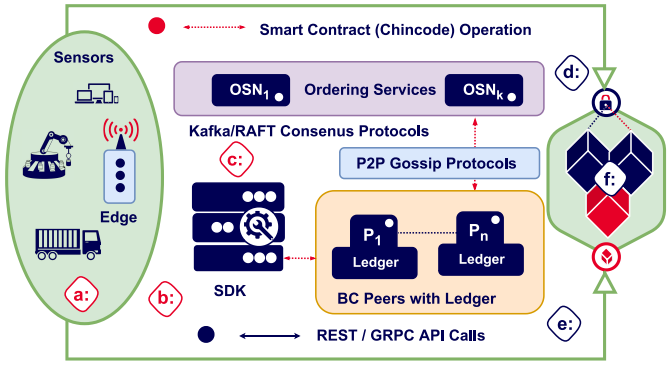

Fig. 7. CBC (hyperledger fabric) data transaction (T_x), verification, and recording flow. It has six steps, i.e., a: Proposing Tx (registrar & submit Tx), b: Endorsing Tx (run environment & endorse), c: Verifying Tx (validate propsed Tx), d: Aggregating Tx (OSN articulate), e: Committing Tx (broadcast to all), and f: Storing Tx (to ledger & DHT).

Algorithm 2: Secure Data Transfer and Store by BC SC

Input : T_x – IIoT data transactions
L – access control lists
σ – signaturues of the T_x
ID_j – identities of the $j^{'}th$ number of IIoT devices
Y – system parameters /* prime numbers, primitive roots etc */
Output: (V_{I_d}, V_{T_x}, S) – set & return verification and storing flag *true*

```
create := (ID, L, Tx, σ, ADS)          /* creates Tx  */
signTx (Tx, Sk)                        /* sign Tx */
castTx (Tx, σ)                         /* broadcasts Tx */
for Tx ← Txi                           /* for all n × Tx */
do
    V1 ← verID (ID, Pk, Y)             /* verify (ID) */
    V2 ← verTx (Tx,ID, Pk, σ)          /* verifies (Tx) */
    if (V1 || V2) then
        S ← storeDHT (Tx, ID):         /* store Tx */
    end
end
```

2) can the signed transaction (T_x) be verified. Fig. 5 illustrates the following steps from 1 to 6.

1) Propose: Client edge sensors initiate the process by registering the devices to the blockchain. It constructs the encrypted transaction proposal (t_x) using (s_k) and invoke the smart contract and software development kit (SDK).

2) Endorse: SDK requests for endorsement, and BC peer verifies t_x after authenticating the *id*.

3) Verify: The verification requires meeting the policy, i.e., business logic. The SC takes t_x as input and returns a multi-signed 0 or 1 in response to the SDK apps. T_x is determined as query function using APIs (i.e., OAuth 2.0 REST API). In either case, the SDK apps proceed t_x with the required operations, such as *create*, *retrieve*, *update*, and *delete* with the endorsement.

4) Aggregate: The SDK apps aggregates all consents into single transaction and disseminates those to the ordering service node (OSN). The OSN works on the consensus protocols, i.e., practical Byzantine fault tolerance (PBFT) within Apache Kafka platform.

5) Commit: t_x then relayed to the OSN, associated channel peers confirm each t_x of the block by specific smart contract and checking through concurrency control cersion (CCV). In case any transaction misses the process is identified with an invalid status inside that block. Hence, a fresh block is committed to the BCL.

6) Store: The Gossip protocol of the OSN broadcasts ledger update across the BC network. Thus, pointer is memorized in the ledger, and data address is securely stored on the offline or online DHT data repository upon IPFS or HL CouchDB. Here, the signature algorithm can be represented as a triple/4-tuple of probabilistic polynomial-time algorithms (G, S, V) or (G, K, E, D) that includes generation (G), *signing*(S), verification (V), key distribution (K), encryption (E), and decryption (D), respectively. Besides the identities ID_j, here, the devices require the access control list (ACL) before transaction (Tx) creation and signing (σ). The industry 4.0 devices along with the edge gateway are solely responsible to create the ACl list (L) in addition to signature (σ) generation and transaction (T_x) publishing. However, the same L will be required later to access data. The algorithm as shown in Algorithm 2 outcomes three different flags (V_1, V_2, S) set after successful execution. If the identities belong to the derived public keys, $V_1 :=$ true, while the certificateless signature meets the condition as discussed earlier, ($V_2 :=$ true). The blockchain peers do the transaction (T_x) verification in response to the reception. Interchangeable verification procedure works in case of data accessing. Similarly, after data are written to the DHT, the third flag gets set ($S :=$ true). Finally, a new block is added to the blockchain and subsequently the ledger gets updated including the T_x Pointer (Tp).

IV. Data Set and Model Training

There are two fundamental steps that are applied during the data preparation process based on the combined data set applied [17], [18]. Some features, such as—date, time, and timestamp, have been omitted from feature vectors as they may cause to overfit the training data. Furthermore, for some DL and DTL models, the input data shape has been reshaped into three dimensions to feed the models by applying *numpy.reshape* with *swapaxes* and *concatenate* methods. As this dataset originates from multiple heterogeneous sources. So, it is an essential step to combined all the IoT sensors' data by *redundancy* and *correlation* analysis, which has been evaluated using *Pearson's product-moment coefficient* equation [23]

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where n is the number of tuples, x_i and y_i are the respective values in tuple i, and $\bar{x}$ and $\bar{y}$ are the respective mean values of x and y.

As datasets originate from different heterogeneous sources, it is an essential step to combine all the IoT sensors' data by *redundancy* and *correlation* analysis. This analysis has measured how strongly one feature, i.e., *door_state*, implies the other, i.e., *light_status*. Table II shows the respective correlation analysis where light status (LS), door state (DS), Smartphone signal (PS), temperature condition (TC), pressure

TABLE II
CORRELATION MATRIX (WITH COLOR INTENSITY)

LS	DS	PS	TC	PR	CT	HY	TE	
0	0	0	0	0.12	0	0.01	0.15	**TS**
0	0	0	0	0.02	0	0	0.01	**MS**
0	0	0	0	0.02	0	0	0.03	**LG**
0	0	0	0	0.02	0	0	0.03	**LT**
0	0	0	0	0.57	0.16	0.35	0.60	**FT**
0.01	0.16	0.01	0.01	0	0.58	0		**TE**
0	0.01	0	0	0.04	0.33			**HY**
0	0	0	0	0.55				**CT**
0	0.12	0	0.01					**PS**

TABLE III
DTL PERFORMANCE COMPARISON METRICS

Algorithm	Accuracy	Precision	Recall	F1Score	ROC AUC
FCN	0.84	0.85	0.84	0.83	0.81
LeNet	0.80	0.82	0.80	0.79	0.76
IncepNet	0.80	0.86	0.80	0.81	0.73
MCDCNN	0.80	0.83	0.80	0.79	0.76
CNN	0.81	0.83	0.81	0.80	0.76
LSTM	0.85	0.84	0.77	0.77	0.83
MLP	0.73	0.74	0.78	0.77	0.74
ResNet	**0.87**	**0.88**	**0.86**	**0.86**	**0.83**

(PS), current temperature (CT), humidity (HY), and temperature (TE) are in the rows. Similarly, thermostat status (TS), motion status (MS), longitude (LG), latitude (LT), fridge temperature (FT), temperature (TE), humidity (HY), and current temperature (CT) are in the column, respectively. We have performed these strategies to scale the selected feature values within a range between [0.0, 1.0] using a technique called *minimum–maximum normalization* [24]

$$N_{\text{ormalized}}V_{\text{alue}} = \frac{(X - X_{\min})}{(X_{\max} - X_{\min})}$$

where X is an original value and $X_{\max}$ and $X_{\min}$ are the maximum and minimum values of the feature, respectively.

A. Training the DTL ResNet Model

First, we have divided the combined data set into the training data set (80%) and the validation data set (20%) by using the *train_test_split* method of the *scikit_learn* library. To avoid the overfitting problem, this splitting ratio has been considered as the best ratio between the training and the validation data set [25]. We have used the value of the *random_state* parameter as true (1), which decided the splitting of data into the training and the validation set randomly. The *k*-fold *cross_validation* has been used for parameter tuning.

V. APT DETECTION EVALUATION

First, we consider the quantitative performance of DTL algorithms. Table III shows the quantitative performance summary of the DTL algorithms, where the proposed ResNet model shows an optimal performance compared to the other DTL algorithms with an accuracy score of 0.87, precision score of 0.88, recall score of 0.86, $f1$-score of 0.86, and ROC AUC score of 0.83. In this model, we used three hidden layers where *relu* is the hidden layer activation function. Also,

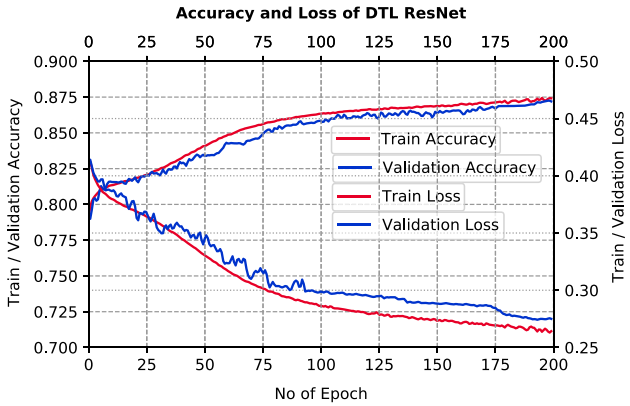

Fig. 8. Training and validation accuracy of the ResNet model used to detect APT. The experiment performed within an edge complaint setup.

softmax is used as a network output activation function, and "categorical_crossentropy" is used as a loss function along with *adam* optimizer.

Fig. 8 shows the accuracy score of every single epoch for both training and validation phases on the above-mentioned eight discrete DTL models. Considering epoch number 170 to 200 for both phases, the flattening characteristics of the curve and accuracy are not increasing literally, we have considered 200 epochs for our analysis. The proposed ResNet model (P-ResNet) shows the highest accuracy score in both the training and validation phase. Whereas, MLP model shows the lowest accuracy score during any settings of the epoch number within 1–200. In detail, according to Fig. 8, the accuracy of the MLP model starts around 0.67 for the training phase and 0.63 for the validation phase in the epoch number 10. But it increase dramatically around 0.78 in epoch number 65 and 0.74 in epoch number 140 for the training and validation phase, respectively.

However, for the training phase, the accuracy score (0.78) remains stable in epoch number 66 to 200. On the other hand, for the validation phase, the accuracy score (0.74) remains stable between epoch numbers 141 and 200. The training and validation accuracy of CNN, IncepNet, LeNet, and MCDCNN models remain steady between the epoch number 1 and 200 as shown in Fig. 8. The accuracy of LSTM and FCN models starts with a score of 0.82 at the beginning. But this score rises gradually with the increase if the epoch number and reaches to approximately 0.86 when the epoch number is 160 and then remains stable between epoch number 161 and 200 for both of the phases. The remarkable point is that the behavior of the training phase almost similar to the validation phase. Fig. 8 shows the trend of the accuracy score of both phases for a better understanding of our proposed model. The accuracy of the proposed model jumps rapidly in epoch number 60 and it reaches a peak of point close to 0.87 at epoch number 169. However, according to Fig. 8, which remains almost stable up to the early stopping checkpoint with an accuracy of 0.87.

VI. BLOCKCHAIN PERFORMANCE EVALUATION

A. Platform Setup

The proposed security approach was deployed inside the Caliper evaluation toolkit for IBM Hyperledger Fabric (HLF) v1.4.1. It helps measuring a particular blockchain

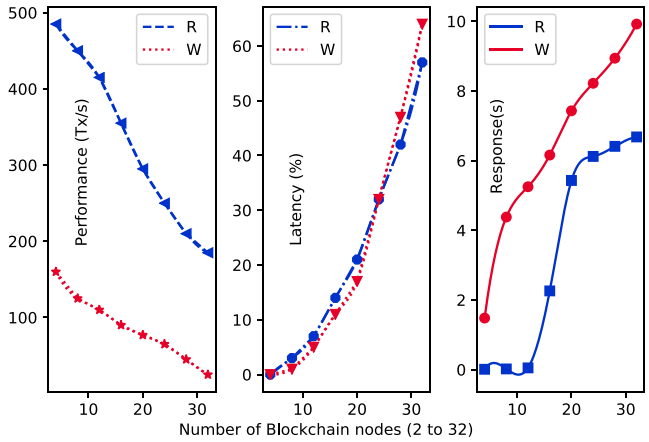

Fig. 9. System performance based on blockchain transaction processing rate. Here, (a) performance, (b) latency, and (c) response time for the use case with 2–32 nodes.

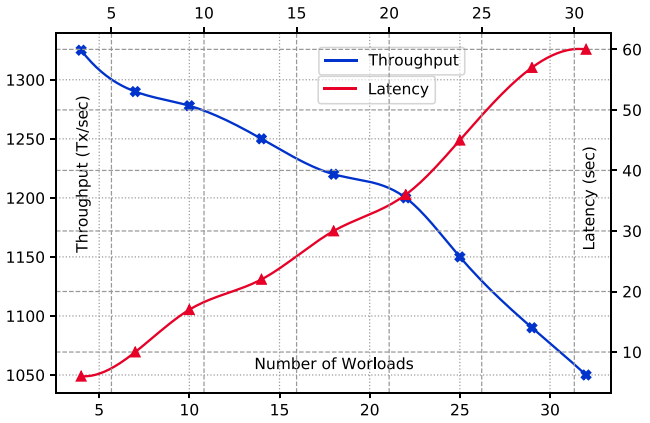

Fig. 10. Proposed system scalability. It shows performance based on an average *throughput* versus *latency*.

deployment with a set of previously defined enterprise use cases. IBM discloses that no general tool provides performance evaluation benchmark for blockchain while releasing the initial version of HLF Caliper [7]. The integrated use cases were customized to overlap the industry 4.0 edge requirements for generating data. However, the latest version of the Nodejs package manager (NPM 8.0.1), docker, and curl was installed to set up the runtime environment inside the Ubuntu 18.04 LTS with 16 GB of memory where *python2*, *make*, *g++*, and git ensure additional SDK supports. A typical configuration for the permissioned blockchain has programs called *Test Harness* that include client generation and observation and the deployed blockchain system under test (SUT) and the RESTful SDK [7].

B. Blockchain Deployment

The RESTful SDKs interface among the required components setup. There are four steps required to evaluate the performance benchmark, such as: 1) starting a local Verdaccio-server for package publishing; 2) the connecting the repository to the server; 3) installation and binding the command line interface (CLI) from the server side; and 4) hence, running the integration benchmark. The associated ledger works with the initial *config.yml* file on CLI. After the initial configuration, the system was configured for performance benchmark with the tasks, such as—1) invoke policy checking functions (READ) and WRITE *Tx* into the ledger; 2) setup multiple test cases for about 2–35 number of peers representing industry stakeholders and cosigners; and 3) allocating workloads from 100 to 1500 Tx/sec among those peers representing the Industry 4.0 edge data population. However, the future scope of this work includes increasing the workload to best suit the higher Industry 4.0 CPS standard.

C. Performance Analysis

The Caliper benchmarking results illustrate the deployed project performance based on four measurement metrics: 1) success rate (ρ); 2) latency (Δt and L); 3) throughput (P); and 4) resource consumption (W) for different test cases. Fig. 9 shows the throughput, success rate, and delay, respectively. The associated test case was run under different numbers of workload (W) ranging from 100 to 1000 workloads. The HLF network occupies two chaincodes, four peer nodes, and three OSNs running on Apache Kafka for the PBFT consensus algorithm. As seen in Fig. 9(a), the WRITE has 185 at about 200 workload (W) with the maximum success rate of 93% and an average delay of 5 s. On the other hand, Read operation seems to have a maximum of 470 throughputs on a similar success rate at the maximum workload. The usual delay appears to be almost half of the W delay as W has to incorporate OSNs on Kafka.

The benchmark evaluation explicitly illustrates that the setup configured has lower performance for higher number workload (W) though the theoretical solution proves the CBC has significant adaptability for higher number of nodes. As investigated deep inside, the local workload processing bottleneck affects throughput and latency. Hyperledger T_x flow works demand enough responses against the submitted T_x proposals, in case the responses are queued due to network overhead, bandwidth, or processing loads consequences the latency raising. On top of that the general purpose workstation configuration is slower the evaluation for higher workloads. Here, Fig. 10 portrays the relation between performance and scalability based on the previously executed Read and Write Operations. To avoid further complexity, OSN and peer configuration left resembling to initial setup. However, two test cases run for 300 and 500 workload. As depicted by Fig. 10, the HLF platform setup has lower scalability. For the first test case (300 workload), the throughput and latency, respectively, reach 150 tps and 64. However, for the other test case, it comes with lower throughput and higher latency with respect to the number of nodes ranging from 4 to 32. Caliper toolkit allows to run the node subset that endorse particular chaincodes. The investigation shows that the proposed technique without a certificate can respond within 1–16 ms. However, it delays 40–242 ms with the default CA of the Hyperledger CBC deployment. The response latency varies with the increase of workloads.

VII. Conclusion

The security of the critical Industry 4.0 CPS deserves immense concern as any leakage should outcome devastating financial damages and loss of lives. On top of malware, ransomware advanced persistent threat (APT) has been responsible for such loss. Industry 4.0 edge ecosystem wonders for

a cooperative trust building rather than trusting a single entity that the proposed security technique purposely promises to offer through a certificateless mechanism. Admittedly, an inadequate data-protection mechanism can readily challenge the security and reliability of the network. Considering the detection accuracy, the proposed approach has utilized the salient features of the DTL algorithm upon the ResNet model. After successfully filtering the APT from the edge end, that data are transferred to the associated DHT storage. CBC network ensures the IoT sensor registration, authentication, and validation. The immutable ledger records the data and APT detection transaction. The proposed detection model has an overall accuracy score of about 90% where CBC increases the data transaction rate.

REFERENCES

[1] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9463–9472, Jun. 2021.

[2] X. Wang, S. Garg, H. Lin, M. J. Piran, J. Hu, and M. S. Hossain, "Enabling secure authentication in industrial IoT with transfer learning empowered blockchain," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7725–7733, Nov. 2021.

[3] P. Zhang, H. Sun, J. Situ, C. Jiang, and D. Xie, "Federated transfer learning for IIoT devices with low computing power based on blockchain and edge computing," *IEEE Access*, vol. 9, pp. 98630–98638, 2021.

[4] Z. Rahman, I. Khalil, X. Yi, and M. Atiquzzaman, "Blockchain-based security framework for a critical industry 4.0 cyber-physical system," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 128–134, May 2021.

[5] Y.-P. Lin and T.-P. Jung, "Improving EEG-based emotion classification using conditional transfer learning," *Front. Human Neurosci.*, vol. 11, p. 334, Jun. 2017.

[6] S. T. Mehedi, A. Anwar, Z. Rahman, and K. Ahmed, "Deep transfer learning based intrusion detection system for electric vehicular networks," *Sensors*, vol. 21, no. 14, p. 4736, 2021.

[7] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "GDPR-compliant personal data management: A blockchain-based solution," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1746–1761, 2020.

[8] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *Proc. IEEE*, vol. 108, no. 10, pp. 1825–1848, Oct. 2020.

[9] L. Kiffer, R. Rajaraman, and A. Shelat, "A better method to analyze blockchain consistency," in *Proc. 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain*, 2018, pp. 729–744.

[10] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Comput. Surveys*, vol. 53, no. 6, p. 126, Dec. 2020.

[11] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 931–948.

[12] G. De La Torre Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662.

[13] X. Lin, J. Wu, A. K. Bashir, J. Li, W. Yang, and J. Piran, "Blockchain-based incentive energy-knowledge trading in IoT: Joint power transfer and AI design," *IEEE Internet Things J.*, early access, Sep. 15, 2020, doi: 10.1109/JIOT.2020.3024246.

[14] X. Jiang, F. R. Yu, T. Song, Z. Ma, Y. Song, and D. Zhu, "Blockchain-enabled cross-domain object detection for autonomous driving: A model sharing approach," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3681–3692, May 2020.

[15] H. Lin, J. Hu, X. Wang, M. F. Alhamid, and M. J. Piran, "Toward secure data fusion in industrial IoT using transfer learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7114–7122, Oct. 2021.

[16] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.

[17] N. Moustafa. "The Bot-IoT Dataset." 2019. [Online]. Available: https://ieee-dataport.org/documents/bot-iot-dataset

[18] N. Moustafa. "Ton_IoT Datasets." 2019. [Online]. Available: https://ieee-dataport.org/documents/toniot-datasets

[19] M. Billah, A. Anwar, Z. Rahman, and S. M. Galib, "Bi-level poisoning attack model and countermeasure for appliance consumption data of smart homes," *Energies*, vol. 14, no. 13, p. 3887, 2021.

[20] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[21] Y. Miao, C. Chen, L. Pan, Q.-L. Han, J. Zhang, and Y. Xiang, "Machine learning-based cyber attacks targeting on controlled information: A survey," *ACM Comput. Surveys*, vol. 54, no. 7, p. 139, Sep. 2022.

[22] J. Zhang, L. Pan, Q.-L. Han, C. Chen, S. Wen, and Y. Xiang, "Deep learning based attack detection for cyber-physical system cybersecurity: A survey," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 3, pp. 377–391, Mar. 2022.

[23] J. D. Chee, *Pearson's Product-Moment Correlation: Sample Analysis*, vol. 4, Research Gate, Berlin, Germany, 2015, pp. 4–90.

[24] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 621–636, 2018.

[25] I. Guyon, *A Scaling Law for the Validation-Set Training-Set Size Ratio*, vol. 1, AT&T Bell Lab., Holmdel, NJ, USA, 1997.

Ziaur Rahman received the graduate degree from the Shenyang University of Chemical Technology, Shenyang, China, and the master's degree from the Islamic University of Technology, OIC, Gazipur, Bangladesh.

He is a Doctoral Research Scholar of Cybersecurity with RMIT University, Melbourne, VIC, Australia. He served the Department of ICT, Mawlana Bhashani Science and Technology University, Tangail, India, as an Associate Professor. He casually served RMIT University; Monash University, Clayton, VIC, Australia; Deakin University, Geelong VIC, Australia; and Charles Sturt University, Bathurst, NSW, Australia. He coauthored three articles, which were nominated and received the best paper awards. His research interests include blockchain technology, security of the Internet of Things, and machine learning.

Mr. Rahman is affiliated with the IEEE, ACM, and Australian Computer Society.

Xun Yi received the Ph.D. degree from Xidian University, Xi'an, China, in 1995.

He is currently a Full Professor of Cybersecurity with the School of Computing Technologies and the School of Science, RMIT University, Melbourne, VIC, Australia. He has published more than 200 research papers in international journals and conference proceedings. He has ever undertaken a program committee members for more than 30 international conferences. Recently, he has led some Australia Research Council Discovery Projects in Data Privacy Protection. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and data privacy protection.

Prof. Yi was an Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING from 2014 to 2018.

Ibrahim Khalil received the Ph.D. degree from the University of Berne, Berne, Switzerland, in 2003.

He is currently a Full Professor with the School of Computing Technologies, RMIT University, Melbourne, VIC, Australia. Before joining RMIT University, he also worked with EPFL, Lausanne, Switzerland; the University of Berne, Bern, Switzerland; and Osaka University, Osaka, Japan. He has several years of experience in Silicon Valley-based companies working on large network provisioning and management software. His research interests are in scalable efficient computing in distributed systems, network and data security, secure data analysis, including big data security, steganography of wireless body sensor networks, and high speed sensor streams and smart grids.