Software-Defined Security-by-Contract for Blockchain-Enabled MUD-Aware Industrial IoT Edge Networks

Prabhakar Krishnan, *Member, IEEE*, Kurunandan Jain, Krishnashree Achuthan, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—To ensure the proper functioning and performance of Industrial grade Internet of Things devices (IIoT) in Industry 4.0 networks, it is critical to identify the capabilities and malfunctions of their component devices (e.g., sensors, actuators, and controllers) and detect potential misbehavior arising due to cyber-attacks, and misconfiguration. We envision future IoT devices embed behavioral profiles through *Security-by-Contract* (S×C) that are easy to validate and verify against network security policies; manufacturers to provide manufacturer usage description (MUD) profiles as a *manifest* for the devices to signal to the network what sort of access and network functionality they require to properly function. We design authentication in the IoT onboarding process, employ blockchains to a verifiable and immutable repository to store this network manifests, that is signed and verifiable with S×C based *smart contracts* by the device manufacturer, or industry authority. The integrated framework combines blockchains and S×C security contracts, MUD-based behavioral fingerprinting, and software-defined-networking for managing the security of IIoT ecosystems. Finally, the proposed scheme is validated in a simulated IoT environment on various performance parameters.**

***Index Terms*—Behavioral Compliance, blockchain, industrial Internet of Things (IIoT), network security, software-defined-networking (SDN).**

I. Introduction

INDUSTRIAL Internet of Things (IIoT), including devices, computers, and intelligent embedded devices that allow smart operations, are significant enablers of Industrial 4.0 [1]. Industrial mission-critical Internet of Things (IoT) has become a leading device model in modern smart cyber-physical systems (CPS), the failure of which can lead to substantial economic and operating costs. It is simply a network of smart devices predesigned and preconfigured to perform automated and recurring, scalable, in-press, secure tasks through critical data collection and protocols of dissemination, such as CoAP, MQTT, and HTTP. The goal-oriented critical insights provided by the IoT platform will enhance management and operations. As the enterprise embraces IoT for industrial usage and many home-based automation solutions, IoT systems are becoming the target for attackers. Attackers are increasingly exploiting vulnerabilities of many IoT systems to steal data generated by IoT sensors, gaining control of devices to perform malicious activities, or just take down the network for fun. Apart from security, privacy concerns are raised with the growing usage of IoT applications. Although some IIoT solutions have emerged from the conventional wireless sensor network, it has not been able to carry out vital tasks in the IIoT industry. Generally, sensor networks are application-oriented silos that lack dynamic configurability. The problem lies in handling heterogeneous data gathered from various sources. Software-defined-networking (SDN) offers a cost-effective solution for improving the agility and flexibility of sensor networks. The softwarized infrastructure also benefits from the programmability of commercial-off-the-shelf hardware to perform networking functions and end-to-end services. As the adoption of high-tech devices, gadgets and connected applications are rapidly increasing in the Industrial ecosystem, this will lead to daunting challenges in security, privacy, and safety of devices, users, and data accessed by the applications [2]. The exploitation of the new generation of mobile and smart IoT devices [3] has shown that these modern technologies can be breached, and the device is tampered with. These security concerns have prompted International standard consortiums to advocate guidelines to both industry and academic communities for building secure IIoT devices and services ecosystems. There are several efforts underway to enumerate and manage IoT devices securely, Cisco's manufacturer usage description (MUD) [4] is one such effort. MUD is standardized by IETF as an extension to DHCP that includes a URI to the IoT device's manufacturer. The device downloads the file (a manifest) that defines the behavioral profile and functionalities. As IoT devices typically serve predefined roles and exhibit predictable behaviors in IIoT environments, we can define MUD profile(s) as a *manifest* formally and succinctly.

Manuscript received March 1, 2021; revised April 19, 2021; accepted May 13, 2021. Date of publication June 2, 2021; date of current version July 11, 2022. Paper no. TII-21-1021. *(Corresponding author: Prabhakar Krishnan.)*

Prabhakar Krishnan, Kurunandan Jain, and Krishnashree Achuthan are with the Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri-Campus, Kerala 690525, India. (e-mail: kprabhakar@am.amrita.edu; kurunandanj@am.amrita.edu; krishna@amrita.edu).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3084341.

Digital Object Identifier 10.1109/TII.2021.3084341

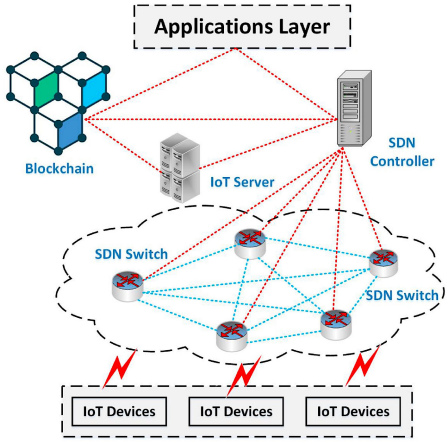

Fig. 1. Conceptual secure IoT design through blockchain and SDN.

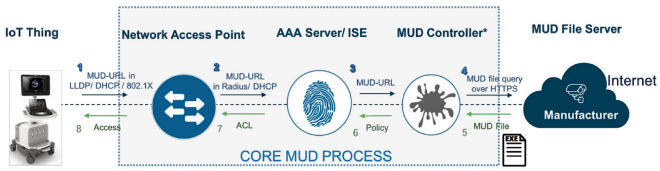

Fig. 2. MUD enabled IoT workflow.

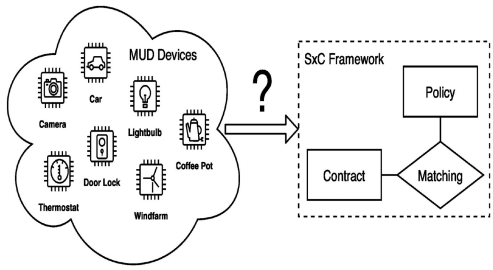

Fig. 3. Integrating MUD and S×C

In an industrial network, when an IoT device is connected to a network, its network manifest is identified by perimeter security mechanisms, and a series of security functions in a chain are deployed to enforce the *least privileged* network access to that device. The drawback of this approach is that the protection tool has no way of knowing what access the IoT devices should be given. If an attacker obtains access to the IoT devices, they may change the manifest, preventing the device from accomplishing its mission. If we had a "*verifiable, immutable, and derived scheme*" to store these *network manifests*, what would happen? To this end, *Smart Contract* distributed ledger technologies are proposed. Contracts among connected devices can be registered on the blockchain ledgers [5], as well as executed automatically to boost transaction performance. The IoT nodes can broadcast the references to the smart contracts during the 802.1× bootstrapping and onboarding phase. The *network manifest* could be securely fetched from the cloud services that are signed/verified by the vendors and trustworthy third-party authorities. A security-by-contract (S×C) or policy data structure [6] is generated by the end-user domain administrator to ensure that the smart contract references are not forged or invalid. This multiparty collaborative security scheme would guarantee greater security and facilitate programmable access control for the Industrial Internet environment. The contract is uploaded to the user's device, and the Edge Controller verifies the contract. The compliance to the operational policies will be verified for each user's device to accept or raise an alert of a violation. Confidential third parties and operational managers could write S×C policies and compliance templates. The S×C framework shows great promise for heterogeneous IIoT networks in terms of sustainable security for devices to exchange information, execute transactions, but in its current state, it requires drastic additions to the development and manufacturing processes of IoT devices.

As shown in Fig. 1, blockchain and SDN can provide secure data transmission in the overall networking system, where the SDN controller manages network services and blockchain provides security and integrity. The network comprises *things* or devices (e.g., sensors, actuators), hubs, switches/routers, servers, and applications, which are interconnected in a peer-to-peer (P2P) and Cloud (WAN) network. As large enterprise networks and critical infrastructural service providers are increasingly deploying *Things* at scale, there is an urgent need for programmable and reliably enforceable security schemes. The MUD-aware IoT is illustrated in Fig. 2. To practically implement the MUD-based access-control, we require broader *expressive language,* and richer definitions. The traffic and behavioral patterns are captured and represented succinctly through MUD profiles. These formal representations/profiles are translated to rules and enforceable at run-time in the switches, all these actions (e.g., whitelist/blacklist) through programs/applications.

The recent works [10] [11] [12] have utilized the SDN architecture for MUD-based IoT bootstrapping, administration, security solutions and results are promising. In an ideal world, manufacturers would produce contracts and store them in their devices, and this is not a realistic short-term goal. We have to face thousands of devices on the market that cannot naturally comply with S×C. But a growing number of these devices are compliant with the MUD specification which allows devices to signal to the network their requirements to work properly.

As shown in Fig. 3, we propose a method for integrating MUD-compliant devices with a blockchain-based S×C framework. We show that it is possible to extract basic S×C contracts from the MUD profiles. Even though the resulting contracts are partial, our work paves a promising way for profiling MUD devices and extracting complete S×C contracts. We claim that S×C, combined with MUD profile in Edge security services, can protect the heterogeneous IoT systems in the Industry 4.0 based infrastructures. In addition to access control and security attributes at the networking protocol level, the manufacturers or the operators can define security property in the MUD profiles.

In this research, we have addressed some of these shortcomings of specifications referenced by prior works and advanced the scalability, efficacy, and real-time attack classification capabilities of ML-based network intrusion detection and prevention systems (NIDS) employing MUD extensions and novel redesign of SDN. We leveraged from publications such as [7], which

concluded that the IoT devices in the Industrial environment could show recognizable traffic patterns and properties, that can be exploited with MUD profiles. To this end, we have built a framework to learn anticipated behavioral trends of MUD compliance and security contracts on each IoT device and validate policies. This article focuses on smart contract-enabled blockchain technology and how it can be used to achieve trustworthy access control for the applications in a MUD-aware SDN-based IIoT infrastructure. The key contributions of our research are as follows.

1) A method for integrating MUD-compliant IIoT devices and applications, based on S×C contracts from MUD definitions.
2) Gateway and Controller that can fingerprint behaviors of the devices, detect anomalies, intrusions, and security breaches.
3) Secure bootstrapping is a critical process in the IIoT environment. We proposed a blockchain-based scheme to enable attestation and lightweight authentication.
4) The system with the ability to validate the MUD policies to comply with the policies of the deployed domain.
5) Exploited the latest IETF standard-based MUD scheme to address IoT security configurability and compliance.
6) The devices are preinstalled with self-testing metadata with S×C contracts that can be checked against security policy, configured by the administrators and end-users. In turn, this establishes the traceability of ubiquitous IoT devices.

The rest of this article is organized as: Section II provides the background and an overview of related works. Section III presents our proposed solution framework. Section IV presents the experiments and performance evaluation. Section V concludes this article.

II. Background and Related Work

In this section, we present the preliminary definitions and overview of recent research in IoT security and privacy.

A. Manufacturer Usage Description

A framework is proposed for fine-grained definitions of access control lists (ACLs) and complex policies (See Fig. 2). The major components are as follows.

1) Thing/device with MUD URI and base policy.
2) Router or Switch/Gateway.
3) MUD Manager that orchestrates the operations.
4) MUD File Server with the MUD profile database.

The MUD manager is hosted at the end-user infrastructure and it validates the signatures from device profiles and configures the router/IDS with the ACLs specified by the manufacturer, translates the MUD entries into standard IDS rules or filter expressions or flow rules for specific network configurations.

B. S×C Fundamentals

The IoT device can specify how it behaves when the data changes through a profile and a contract. The contract has to be shown to the controller before a device joins the network. Safety

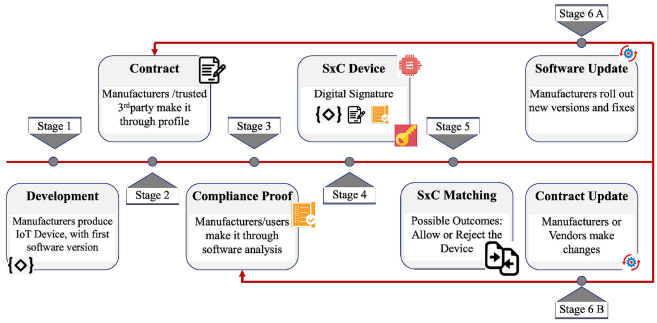

Fig. 4. S×C workflow.

TABLE I
S×C Representation

Device	Actions	Implications
User	Connects the S x C device to the network	Completely Transparent for the User
Administrator	Manages the S x C network Policies	Small Overhead for Policy Management
Manufacturer	Write device Contract, provide MUD, PoC	Overhead for Contract, MUD and PoC Creation

policies or protocols are definitions of appropriate activities for IoT devices regulated by a controller. It is not clear if contracts and practices can be completely understood without recognizing the security activities and behaviors of the devices in the environment. When a device joins the network (see Fig. 4), it is queried by the controller, and validation is done before it can be allowed in the domain. As soon as the contract is updated, it goes back to rescanning to ensure compatibility with the network policy. Ideally, the contracts of IoT devices are factory-written/preinstalled, as this helps to clarify who is responsible for providing support. The end-user should be assured that it will run in a networked environment immediately. The manufacturers will be responsible for drafting formal contracts and proof of compliance (PoC). These contracts formally bind the system feature and are updated accordingly as changes or upgrades are carried out. The PoC is a formally signed proof on the device that binds the code on the device to the contract and is verified externally. If the code or the contract changes, the manufacturers are required to update the contracts and new PoCs. To tamper with the device behavior, the adversaries must do forging of PoCs and comply with the terms by signing with the manufacturer's credential as well. Table I gives a summary of the roles, actions, and implications. By using a PoC, the manufacturer can ensure the IoT software stays in compliance with the contract. The contract is extracted (possibly incomplete, even from legacy IoT devices) by a process called profiling [8]. A system can be sandboxed, and the resulting traffic can be analyzed to determine its behavior.

SDN is now widely adopted as the mechanism for managing cyber-security attacks and risk mitigation [9], due to its global view, programmable control, dynamic traffic shaping, which are key for resilient and fast-reacting real-time IoT networks. The architecture also provides flexibility to deploy virtual network security functions virtual network function (VNFs). Hamza *et al.* [10] are the early proposers to employ the MUD profile-based behavioral analysis and anomaly detection scheme in an SDN-enabled IoT environment. The IDS can accept/drop

whitelist/backlist the traffic from specific devices as defined by the vendor or system administrator. The authors [11] developed a tool *MUDgee* that can generate MUD profiles from PCAP trace files, a monitoring framework for formally/semantically validating MUD files and policy verification, in the context of any given Smart CPS network. Many research proposals are published on the topic of profiling IoT device behavior on a network outside the context of MUD/S×C. The NIST has been recommending the MUD to the industry as an efficient approach [13] that could automate the threat/risk detection process, protect in-secure devices, and remediate from attack, in highly dynamic IoT networks. Through this research, we have attempted to answer some of the open questions in these emerging paradigms to choose the best approach in securing the IoT devices.

C. Blockchain Smart Contract

Smart contracts work well for time-sensitive items, but record-keeping and accountability of distributed ledger are critical. They allow fragmented and decentralized data collection and product tracking. Blockchain can be deployed into *permissionless* or *permissioned* modes. Permissioned blockchain (that is implemented in this research) is not fully P2P. The members involved in the blockchain ledger could exercise more control throughout the transaction process, with higher overall throughput for their transactions. The mining operation is done in the Edge *Gateway* switches that have more computing/storage power than the resource-constrained *things*. The resource-limited IoT applications would never end up being completed. The mining is much less resource-intensive when compared to a permissionless blockchain. The edge switches are in charge of tracking transactions, checking current blocks of transactions, and adding new ones to the blockchain. These IoT devices are just blockchain or smart contract actors. The S×C-aware devices and blockchain systems with preinstalled smart contracts will have access to decentralized applications with assistance from the Edge switches. They just serve as authoritative agents for IoT devices. Each IoT device's gateway makes efforts to store (for further scrutiny) the manifests for the device network and then manage the deployment process of the blockchain on behalf of the IIoT devices. BlockTDM [14] is a blockchain-based trusted metadata-sharing scheme that provides mutual authentication among peers. The framework intends to improve data privacy and security for Edge networks or IoT users. Trustlist [15] incorporates SDN and blockchain technologies to make IoT network management automatic. Thus, the popular design would create the trustworthiness of IoT devices. Blockchain technology is deployed for IIoT device network to verify their identity in the SDN-enabled pervasive edge computing environment [16] for enhanced network security. An edge cloud and SDN-enabled distributed security platform is introduced in [17]. An intrusion detection system is deployed at the cloud layer and is subsequently reduced at the edge layer of the IoT network. The SDN-enabled gateway [26] provides dynamic network traffic flow management that contributes to the security attack identification and security attack prevention by determining dubious network traffic flows and hindering doubtful ones. Poorly controlled devices and faulty firmware updates could compromise a growing IoT network. This article [18] identified a blockchain-based approach to registering firmware updates on IoT devices. This article [19] analyzes the problem of distributed and trustworthy access control for the IoT and proposes a system involving multiple access contracts to solve the problem. EdgeChain [20] uses a "*credit-based resource management*" framework to monitor the resource usage of IoT devices, based on static rules (e.g., "priority, application type, access pattern history"). Smart contracts can automatically govern communication between a connected device and its owner with a predictable and determinate result. All IoT [including device-to-device (D2D)] transactions are registered and logged into a blockchain so that they are secure and auditable. Zhu [21] has developed and demonstrated using blockchain technology for identity management and decentralized access control.

III. Proposed SDN-Enabled Architecture

The SDN/NFV framework comprises functions for enforcing compliance to Security Contracts/Policies and Anomaly Detection based on MUD behavioral manifest in the Gateway and ML-based Classifier/analytics in the Controller. See Fig. 5, for the big picture. The framework is proposed with five layers—

1) Device/SmartThings layer that consists of industrial IoT devices that can sense their ambient environment and interact with IoT Gateway Hub.
2) Gateway/Data Layer, which after receiving the unprocessed and unstructured data from IoT nodes, converts the data to digital stream and preprocesses it so that the data is ready for review. Multiple switches connect with different access methods and attach to the controller.
3) Control/Management layer orchestrates the traffic policies and security contracts.
4) Edge/Fog Layer hosts the infrastructure and manages the computing resources and overall security at the network level.
5) The Cloud/Application layer is implemented for further processing, QoS, and analysis.

A. IoT Security Gateway

The IoT and SDN protocols are combined into one special device called a *Hybrid Gateway* switch. This switch runs a modified switch OS on a coordination layer, combining the SDN/IoT functionalities. On receiving a packet, Switches (IIoT Switch) need to make decisions (match-action-rule) if this new-flow or packet (of existing flow) is matching any flow rule and compliant to the MUD behavioral profile/policy to either forward or drop the packet. The applications are implemented as virtual network functions (VNFs) and deployed in the switch. We implemented the MUD extension to DHCP and as well as LLDP and 802.1×, which includes a URI to the IoT device's manufacturer, a translation layer to convert the MUD profile/policies to SDN OpenFlow flow rules. The OpenFlow specification defines the common structure of experimental match fields, messages, and actions, then each vendor can customize the format of each structure. We implemented a vendor type OFPT EXPERIMENTER message and encapsulate custom management messages in standard OpenFlow channel from the controller to dataplane switches.

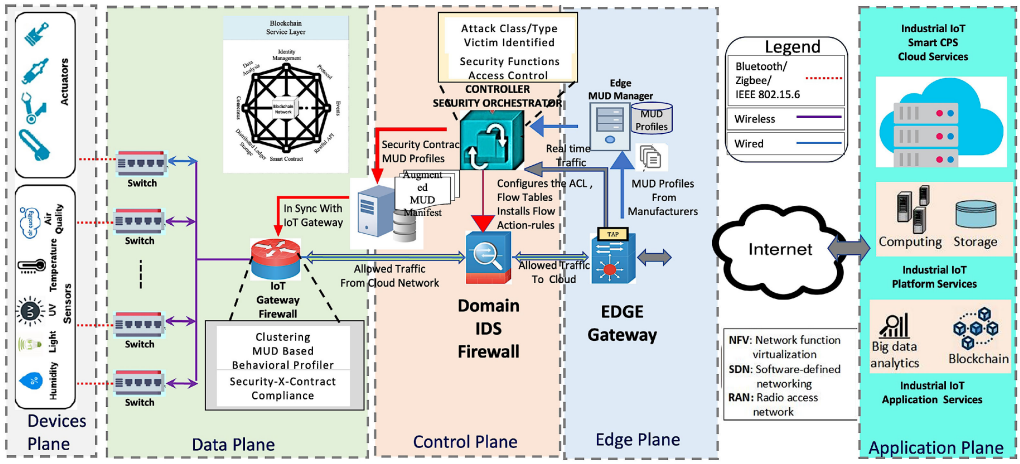

Fig. 5. Proposed software-defined blockchain-based security framework for MUD-Aware industrial-IoT infrastructure.

B. Threat Model

To determine our security and privacy assurances, we create a threat model in which active adversaries can hijack, clone, recover, or even change the information stored in our deployed devices. If adversaries are limited by computational speed, they can launch passive attacks such as eavesdropping, active network attacks. They may also covertly alter or block communications among parties. Additionally, malicious actors can access online private keys and impersonate the identity. We believe that gateways are externally firewalled and therefore stable, so they can be used with confidence. With *permissioned* blockchains, proof of work is no longer a prerequisite for Consensus and Sybil attacks are unlikely.

C. S×C Compliance Verification

We describe the components of the S×C system and the different functions and services for checking the conformance of the S×C-capable devices in the IIoT Edge infrastructure.

Definition 1: Let I be the set of all devices, and a well-formed IoT device given by a specific manufacturer as M_{I} where I is the name of the device and M is the manufacturer

Definition 2: A service name S that can be performed by any of the IoT devices in the set I and the manufacturer M, which can further be expressed by MIS

Definition 3: We denote the non-empty set D to be the list of all possible domains that contain the elements of network domains such as LAN where security rules apply.

For e.g., $D_1 =$ LAN, $D_2 =$ Wi-Fi, $D_3 =$ WAN, etc.

Definition 4: A security rule R is an n-tuple represented by fields listed in Table II. Even though we use only 5-tuples here, this definition is quite general and can be extended.

Definition 5: A security policy, P_G where G denotes a gateway node is a nonempty, nonorder set of security rules defined by the following:

$P_G = \{R_1, R_2, \cdots R_m\}$, where $m > 0$, $\forall_{\mathrm{i,j}} \in R_{\mathrm{i}} \neq R_{\mathrm{j}}$: $i \neq j$

In the blockchain-based S×C approach: A device is accepted in a network governed by a Gateway if and only if the smart contract of the device matches the policy of the MUD manifest and verified with the blockchain ledger. Algorithm 1 (see Fig. 6) shows the consistent verification. Algorithm 2 (see Fig. 7) provides the pseudo code to verify direct D2D communications. Algorithm 3 (see Fig. 8) provides a way of checking if any pair of devices exchange malicious flows between them. In Algorithm 4 (see Fig. 9), we show the logic to validate the S×C contract for a given policy defined at the infrastructure.

With S×C, if the PoC embedded on the device checks out, it can be concluded <*Code, Contract, PoC*> are compliant. The administrator's role is to generate S×C contracts, MUD profiles, policies, maintain the SDN Gateway/Controller.

TABLE II
SECURITY RULE STRUCTURE

Device *I*	The Device Name and Manufacturer M expressed by I_M *M.*+ any device from a specific manufacturer *M.* _. * any device from any manufacturer M.
Domain D	The Doman Name where the security rules apply.
SHARES	List of Devices that any device *I* can interact within the domain *D*
P	List of services that can come from any set $S = \{S_1, S_2, \cdots S_n\}$, where $n > 0$, that the IoT device *I* provides the devices in SHARES[*I*]
RE	List of services that can come from any set $S = \{S_1, S_2, \cdots S_m\}$, where $m > 0$, that the IoT device *I* requires for the function and operation
Rule R	$R[I_i]$ => Device I_i of a security rule *R*. $R[D_i]$, $R[SHARES]$, $R[P]$ and, $R[RE]$ denote the related fields of *R*.

Definition: Security Contract

A security contract denoted as SC_{I} of an IoT device *I* is a non-empty and non-ordered set of security rules concerning all of the devices in the set *I*:

$SC_{\mathrm{I}} = \{R_1, R_2, \cdots R_n\}$,
where $n > 0$ and
$\forall_{\mathrm{i,j}} \in R_{\mathrm{i}} \neq R_{\mathrm{j}}$: $R_i[\mathrm{I}] = R_j[\mathrm{I}]$

A security contract SC_{I} from the set of devices *I* is said to be consistent if these conditions hold:

1) $\forall \mathrm{R} \in SC_{\mathrm{I}}$ is *well formed*
2) $\forall \mathrm{R} \in SC_{\mathrm{I}}$ is *core in* SC_I

Algorithm 1 InConsistentContract

Result: True or False
Require: $SC_{\mathcal{I}}$
for $\forall R \in SC_{\mathcal{I}}$ **do**
 if IsWellFormedRule(R) = *False* **then**
return *False*
 end if
end for
for $\forall R \in SC_{\mathcal{I}}$ **do**
if IsCoreRule(R, $SC_{\mathcal{I}}$) = *False* **then**
return *False*
end if
end for
return *True*

Fig. 6. S×C pseudo algorithm codes.

Definition: D2D Communication Rule

Given 2 devices from the set I
I_n with consistent contract SC_{I_n}
I_m with consistent contract SC_{I_m}
$\forall m,n \in I$, such that $I_n \rightarrow I_m$

We define I_n is allowed to directly communicate with I_m, denoted $I_n I_m$,

if $\forall R_* \in SC_{I_n} R_o \in SC_{I_m}$:
$R_*[RE] \cap R_o[P] \neq \varphi$
$R_*[SHARES] \subseteq R_o[SHARES]$

Algorithm 2 AllowedDirectCommunication
Result: True or False
Require: SC_{I_n}, SC_{I_m}
if DirectCommunication(SC_{I_1}, SC_{I_2},.... SC_{I_m}) = *False* **then**
return *False*
end if
for $\forall R_* \in SC_{I_n}$ **do**
for $\forall R_o \in SC_{I_m}$ **do**
if $R_{*1}[D] \cap R_{o2}[D] \cap R_{*3}[D] \cap R_{o4}[D] \cdots$
and $T_1 \notin R_o[SHARES]$
and $R_*[RE] \cap R_o[P] \neq \phi$ **then**
return *False*
end if
end for
end for
return *True*

Fig. 7. D2D communication compliance.

Definition: D2D Forbidden Rule

Given 2 devices from the set I
I_n with consistent contract SC_{I_n}
I_m with consistent contract SC_{I_m}
$\forall m,n \in I$, such that $I_n \rightarrow I_m$

We define there is a forbidden information flow between I_n and I_m, denoted
I_n no I_m $\exists R_* \in SC_{I_n} R_o I_n \in SC_{I_m}$:
$R_*[RE] \cap R_o[P] \neq \varphi$
$\wedge R_*[SHARES] \subseteq R_o[SHARES]$

Algorithm 3 ForbiddenInformationFlow
Result: True or False
Require: SC_{I_n}, SC_{I_m}
if AllowedDirectCommunication(SC_{I_1}, SC_{I_2},.... SC_{I_m}) = *False* **then**
return *False*
end if
for $\forall R_* \in SC_{I_n}$ **do**
for $\forall R_o \in SC_{I_m}$ **do**
if $R_{*1}[D] \cap R_{o2}[D] \cap R_{*3}[D] \cap R_{o4}[D] \cdots$
and $R_*[SHARES] \subseteq R_o[SHARES]$
and $R_*[RE] \cap R_o[P] \neq \phi$ **then**
return *True*
end if
end for
end for
return *False*

Fig. 8. D2D information exchange.

Definition: Policy == Contract

Given :
a *contract* SC_I of a device I from set I
a consistent *policy* P_G of a Gateway G,

Contract SC_I matches with *Policy* P_G
if the following rule is consistent:

$$P_F \text{ if } P'_F = P_F \cup SC_{I_n}$$

Algorithm 4 Matching
Result: True or False
Require: P_F, SC_{I_n}
$P'_F \leftarrow P_F$
for $\forall R \in SC_{I_n}$ **do**
$P'_F \leftarrow P'_F + R$
end for
if IsConsistentPolicy(P'_F) = *True* **then**
return *True*
end if
return *False*

Fig. 9. S×C based compliance testing.

D. Authentication and Smart Contract Attestation

Initially bootstrapping of all the smart objects (IoT devices) is done, by authenticating and attesting all the devices in the deployment domain with the corresponding manufacturer's MUD profiles. To prevent leakage of secrets, identity, and data, special attestation protocols have to be used. When a new IoT device/system/equipment is onboarded into the network, it attests to its level of legitimacy. It will check whether or not this new device is appropriate. The S×C contract is generated from the MUD and local domain policy manager. For example, a trusted cluster of IoT devices might require each vendor to complete an attestation exchange and obtain a certified MUD/S×C smart contract and profile to qualify for membership. The MUD manager extracts the "mud-signature" from the MUD profile. The validation ensures that only legitimate devices can join the domain. (See Figs. 10 and 11).

The blockchain framework consists of an ACL, each of which implements the access control for a pair of peers, one security contract (S×C), which receives the policy compliance/violation report based on the MUD profiles.

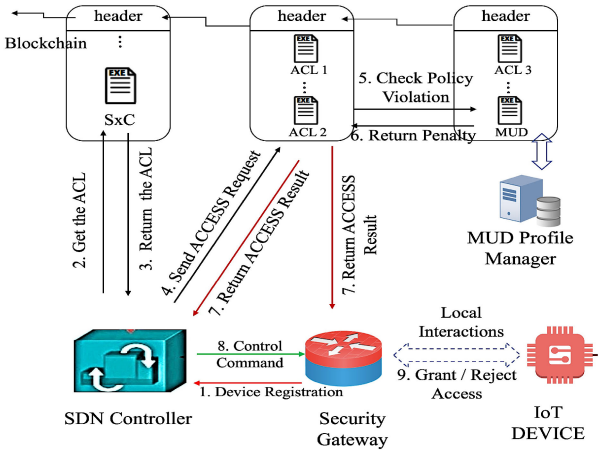

Fig. 10. Blockchain-based smart contract scheme.

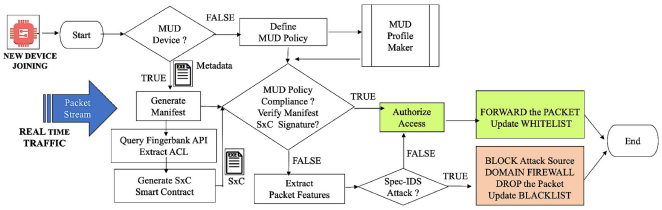

Fig. 11. IoT device joining process.

E. Anomaly and Intrusion Detection

In the IoT Gateway IDS, we employ the outlier detection algorithm for unsupervised anomaly detection. The statistical method helps us to measure the variance of devices' behavior, over a fixed time window, from normal behavior, to detect any anomalies. We used the *Hierarchical clustering* method in our system. Once the different clusters have been recognized, we can test which of the following sets of points are outliers.

We consider the metadata from consecutive packets, i.e., $p^i \rightarrow p^{i+1} \rightarrow p^{i+2} \ldots \rightarrow p^n$ to construct a single feature vector and the packet sequences form reliable semantic data of the sessions. The set of all these characteristic vectors that fits the behavioral profile observed by the system are fed into classifiers. Algorithm 5 shown in Fig. 12 involves creating clusters that have a predetermined ordering from top to bottom for each data point in a three-dimensional vector space. This method is repeated iteratively until all the clusters are merged. We do cross-validation and verification by using anomaly *Z-scores* for outlier detection. The data point instances are sorted based on Z-scores and instance with negative Z-score is flagged as suspicious for further analysis. We augmented our system with dynamically learning models that recompute the profiles in various traffic conditions to classify malicious flows from benign traffic. By computing *"95th, 97.5th, and 99th percentiles,"* the observations that fall outside these boundaries "threshold τ" are marked as "minor outliers." We flag the flows that correspond to outliers beyond this threshold τ and boundary conditions as suspicious and forward them to the controller to be investigated further by the ML-based Classifier system.

Algorithm 5 Outlier Based Anomaly Detection Method

Definitions

Dataset: A dataset X is defined as $\{x_1, x_2 \cdots x_n\}$ with n objects, where each x_i can be categorical attribute represented by a d-dimensional vector, i.e., $X = \{x_{i,1}, x_{i,2} \cdots x_{i,d}\}$.

Pattern Similarity: Two data objects x_1 and x_1 are defined as similar iff (a) $d(x_1, x_2) \ll \tau$ and (b) $d(x_1, x_2)$ 0, if $x_1 = x_2$.

Cluster: A cluster C_i is a subset of a dataset X, where for any pair of x_i's say $(x_i, x_j) \in C$, $d(x_i, x_j) \ll \tau$.

Profile: A profile of a cluster C_i is a mean value, $\mu(x_{\mu,1}, x_{\mu,2} \cdots x_{\mu,d})$ of dataset X , where each $x_{\mu,j}$ is the mean of the j th column of the respective cluster C_i.

Outliers: Data objects, Oi and Oj can be defined as outliers w.r.t a cluster Ci iff (a) Z-score Z(Oi, μi) ≥ τ where μi is the profile of Ci (b) for any other data object Oj in Ci, d(Oi,Oj) > τ.

Proximity measure: each object x_{ij} to a particular cluster in a 3-dimensional space-Euclidean distance, $d(i, j)$

Input

X_p // series of vectors that captures the statistics of packets
X_b// represents statistics of bytes.
M // EM Model

Output

Attack | No Attack | Suspicious

//Calculate mean, variance, Standard deviation (STD)

for all datapoints x_i **do**
Apply M to x_i to obtain probability distribution d
Select maximum probability $d_{i,j}$
$C_i \leftarrow j$
$Q_i \leftarrow d_{i,j}$
end for
for i= 1 to *total clusters* **do**
$P_i \leftarrow Q$ for all Q in cluster i
$S_i \leftarrow$ STD(Q) for all Q in cluster i
end for
//Calculate Z-Scores
for all datapaoints x_i **do**
Apply M to x_i to obtain probability distribution d
Select maximum probability $d_{i,j}$
$Z_i \leftarrow d_{i,j} - P_j$
$Z_i \leftarrow Z_i / S_j$
// Determine outliers based on thresholds
if $Z_i < 0$ then Alert (Suspicious)
else
if $Z_i >$ *Threshold (M)* then Alert (Attack)
else Alert (normal);
end for

Fig. 12. Outlier-clustering based anomaly detection.

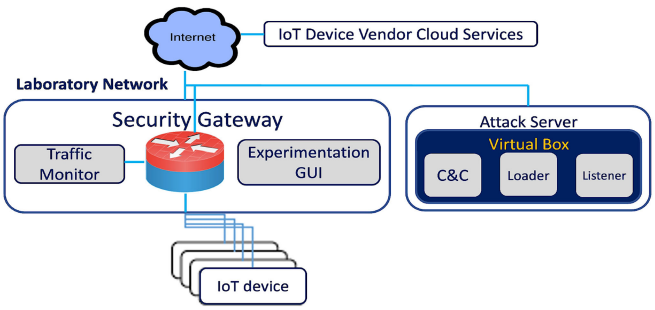

Fig. 13. Laboratory system components.

IV. Performance Evaluation

We implemented our solution on the *Ethereum* platform [22] (see Fig. 13), IoT nodes with "*Raspberry Pi*" and "*Mininet-WiFi*". The hardware platform consists of OvS on *OpenWRT* and *Open Daylight* (ODL) SDN controller software.

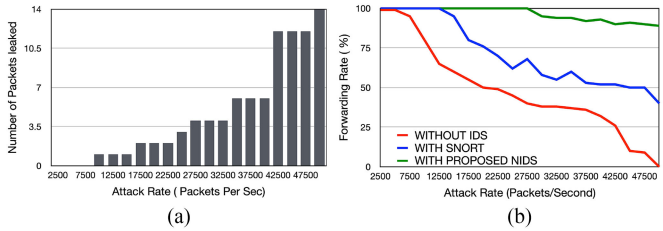

Fig. 14. (a) Attack response. (b) Forwarding efficiency.

TABLE III
MUD Processing

Metric	Value
Model Size	20 MB
Run time features	4.5 KB
Model response time	15 ms
latency with STRICT ACL	270 ms
latency with DELAYED ACL	2 ms

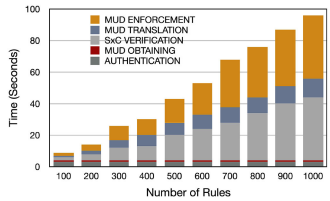

Fig. 15. Overhead in phases.

A. MUD Processing and Enforcement

We measured the response speed of our NIDS during attacks. As the "attempted attack rate" is increased, more packets make it through the firewall before being blocked, reaching a maximum of about 14 packets leakage before the flow rules are applied [see Fig. 14(a)]. Forwarding rate/throughput in the presence of the attack is measured. The usual and attack traffic move through the same switch with legacy IDS "SNORT" loaded in it. Fig. 14(b) illustrates the throughput is sustained with our solution and with SNORT drops to zero.

To quantify processing overhead in our system (MUD behavioral analysis, policy verification), we measure the overhead in TCP connection establishment to a smart object (with and without MUD/ACE) over 100 attempts, with 20% MUD packets, training the classifier with 4 MB for 240 s. With MUD enforcement, new-flow connection latency of 20 ms (Delayed ACE) and 270 ms (Strict ACE), are shown in Table III during attacks. Significantly worse performance for strict ACL is caused by packet drops before flow rules are pushed. The complete end-to-end performance of the system with time consumption for each phase is shown in Fig. 15.

B. Smart Object Authentication and Bootstrapping

Fig. 16(a) shows an extra delay of just 1 s for COAP-EAP bootstrapping. We tested the overhead for the S×C and MUD compliance verification using similar credential/access policies. Fig. 16(b) shows our solution reduced the metadata size by 12%

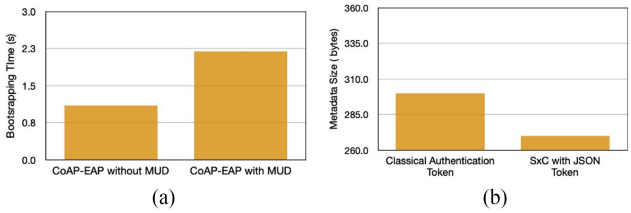

Fig. 16. (a) MUD bootstrapping. (b) S×C-MUD compliance validation.

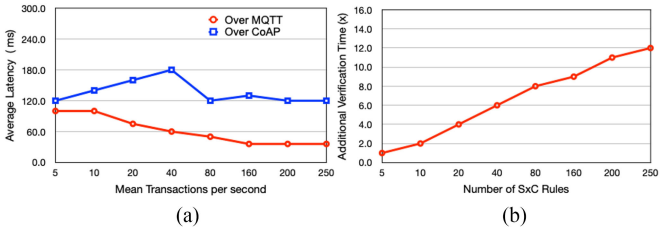

Fig. 17 (a) Registration delay. (b) Verification scalability.

TABLE IV
CLASSIFIER PERFORMANCE

Device	Traffic Detected as			
	Attack		Benign	
	TN (%)	FN (%)	TP (%)	FP (%)
Amazon Echo	98.9	5.0	94.8	1.1
Tplink Switch	96.1	5.4	95.2	4.0
Belkin Motion	88.5	7.1	93.6	7.1
Belkin switch	99.3	7.4	92.9	0.4
LiFX Bulb	99.5	6.3	93.7	0.3
NetAtmo cam	95.6	5.0	95.0	4.4
Hue Bulb	98.2	5.7	92.3	2.3
iHome	99.9	4.1	93.9	0.1
Samsung cam	94.5	5.3	95.3	6.2
Chromecast	89.8	9.2	94.1	9.0

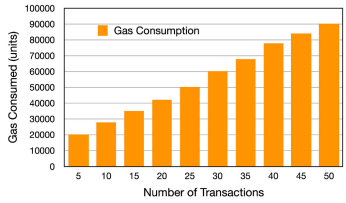

Fig. 18. Ethereum overhead.

(30 B) with JSON-based token. The bootstrapping process is performed only at the time of joining the domain. The average total authentication latency for S×C+MUD processing includes the following.

1) Obtaining MUD profile from sever in IT network.
2) MUD profile translation to SDN OpenFlow policies.
3) S×C compliance checking.
4) MUD enforcement under normal and attack conditions.

When the attack ratio is increased from 0.1 to 0.9, the defense mechanism has to do additional processing and the average bootstrapping time for a node to be added to the network is about 1/9 of that of the open-source SDN scheme. We compared it to the results published by Garcia *et al.* [12], which is a closely related solution. The total time for their smart object (IoT) authentication process is in the order of 4000 ms. But, with our design, since we deploy the MUD-flow rule processing at the data-plane, for a similar count of packet exchanges, the overhead is reduced to 270–2100 ms. The DSA signature generation is slower compared to our scheme and proves we can handle complex/nested network policies.

C. Workflow Performance

IoT Device registration delay: In the Edge, Cloud-IoT protocols such as MQTT and CoAP are utilized. We observed the scalability is unaffected due to the smart contract generation overhead and verification process [see Fig. 17(a)].

Policy Violation Detection: The IoT gateway conducts runtime policy violation identification on sensor data sources. We calculated policy verification speed by varying the number of rules (with a baseline of 5 rules) as configured by the administrator. As shown in Fig. 17(b), the resolution speed is linear with an increasing number of policies.

D. Performance of Anomaly Detection Method

In the Outlier-based Classifier, we utilized *Mudgee* [11] consisting of 28k MUD profile samples and 30 trace files. To avoid malformed entries, we ran MUD profile verification, policy consistency check, and semantic analysis. The datasets from Bot-IoT [23], CICIDS2017 [24], and IoT traffic [25] are utilized to simulate the large-scale attacks. Table IV shows the key classification performance metrics (with attack and benign traffic) of our single-class scheme, which is better than the related anomaly detection solutions [10], [12] from the literature.

E. Blockchain Processing Overhead

The unit *gas* reflects the computational effort required to conduct transactions in Ethereum. Fig. 18 indicates gas consumption per contract and/or transaction. The gas consumption increases with the number of transactions. Our scheme outperformed current strategies by up to 15% in overall transaction throughput and improves the time between transactions by up to 75%.

F. Key Findings and Comparative Analysis

Security has been designed into the system using proven and safe building blocks. We concentrated in this work on the major threats affecting IoT-based infrastructures: Insufficient security configurability, insecure network communications, behavioral anomalies, malicious devices and, security noncompliance. An overview of the salient features in our solution is as follows.

Authentication: IoT devices, nodes, and systems utilize a lightweight crypto signature verification and thus guarantee resistance against spoofing, Unforgeability, and MITM attacks.

Access control: The combined trust in access and credentials is achieved by the use of S×C manifest and validated smart contract. The blockchain ledgers record the control operations and the data can only be accessed by the signed/verified nodes.

Network Security: The combination of device-level (S×C, MUD) communication level (cryptography) and network-level (IDS, secure gateway) security mechanisms provide resistance to recent attacks such as *Cloning*, *Botnet,* and *Phishing*.

Privacy: In contrast to the pseudonymity of other blockchain ledgers (e.g., Bitcoin), our transactions are used only for simple identity operations, and communications with other parties are

TABLE V
COMPARISON WITH OTHER SECURITY PROPOSALS

Reference	General Description	Gaps Addressed in our proposal
Hamza. [10]	MUD files are generated from pcap files to create public profiles.	The generation methodology of MUD files is not considered.
Hamza [11]	Uses MUD/ SDN-based approach to monitor suspicious behaviors and compliance.	MUD file creation, translation in SDN and run-time violation factors are not considered.
Garcia [12]	SDN-based framework for enforcing network access control and mitigating ARP spoofing attacks validated with MUD files.	Configuration and firmware vulnerabilities, no authentication mechanism of the MUD profiles. We solve this through Blockchain attestation.
Giaretta [8]	SDN-based approach with Security-by-Contract.	We extend MUD standards for the IIoT devices, to provide hints so that the controller can pick the appropriate strategy based on the context.
Zhaofeng [14]	Proposed a challenged based collaborative intrusion detection mechanism for establishing trust in detecting insider threats	Although this model builds trust for detecting insider attacks, it does not propose a threat-sharing model and collaborative OT/IT networks.
Kataoka [15]	SDN & Blockchain-based secure and energy-efficient architecture for IoT networks using clustering structure.	Useful for network mapping at a high level but analytics can be cumbersome. In contrast, our approach can re-verify a fingerprint with light weight small metadata synapses and less traffic.
Gao [16]	An innovative, collaborative platform that intent to permit and incentivize parties to interchange network alert data, thereby increasing their overall detection proficiency	The game-theoretic model is complicated and infeasible for threat data sharing. Our distributed and collaborative framework delivers high overall utilization.
Medhane [17]	SVM, Blockchain Raspberry Pi system	IDS has an additional overhead of blockchain. Our inline gateway based Blockchain overhead is within practical limits and optimal trade-off.

off the chain and cannot be traced. Besides, subjects may create as many as partial identities, thus providing enhanced privacy.

Table V shows the summary of the most related works comparing with the proposed work. We observe that there is a potential scope to contribute to the technological aspects, e.g., integration of technologies such as IoT, Blockchain, SDN with MUD standards towards smart and secure IoT applications.

V. CONCLUSION

Through this research, we have demonstrated an advanced autonomic verifiable scheme to build a trustworthy and secure IIoT ecosystem based on S×C. We have evaluated SDN and blockchain technology for the security and privacy of the Industrial IoT ecosystem. S×C paradigm and the most recent IETF standard MUD profiles are integrated to describe the functional requirements of Industrial IoT devices. With this novel scheme, we attempt to improve the configurability, compliance, and behavioral descriptions. The MUD standard is also increasing the scope to explicitly describe the predicted actions and intents of IoT devices. Since the device's information and manifest are stored at the controller, it is critical for privacy by deploying data protection and encryption strategies at the Edge. The data obtained from diverse devices and vendors differ widely, making it a crucial challenge to unified management. The smart contract framework's multiparty authentication would build a highly secure and vastly enhanced IoT infrastructure, also allowing programmable and highly adaptable manufacturer described least privileges for network security.

REFERENCES

[1] G. Aceto, V. Persico, and A. Pescapé, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Comm. Sur. Tut.*, vol. 21, no. 4, pp. 3467–3501, Oct.–Dec. 2019.

[2] F. Loi *et al.*, "Systematically evaluating security and privacy for consumer IoT devices," in *Proc. ACM CCS Workshop IoT S&P*, 2017. pp. 1–6.

[3] D. E. Kouicem *et al.*, "Internet of things security: A top-down survey," *Comput. Netw.*, vol. 141, pp. 199–221, 2018.

[4] E. Lear *et al.*, "Manufacturer usage description specification internet-draft draft-ietf-opsawg-mud-18," IETF Secretariat, 2018.

[5] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[6] A. Giaretta, N. Dragoni, and F. Massacci, "Protecting the Internet of Things with security-by-contract and Fog computing," in *Proc. IEEE World Forum Internet Things*, 2019, pp. 1–6, doi: 10.1109/WF-IoT.2019.8767243.

[7] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2017, pp. 559–564.

[8] F. Matthiasson, A. Giaretta, and N. Dragoni, "IoT device profiling: From MUD files to SxC contracts," in *Proc. Lecture Notes Inf.*, 2020, pp. 143–154.

[9] A. Molina Zarca *et al.*, "Security management architecture for NFV/SDN-aware IoT systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8005–8020, Oct. 2019

[10] A. Hamza *et al.*, "Combining MUD policies with SDN for IoT intrusion detection," in *Proc. ACM Sigcomm Workshop IoT S&P*, 2018, pp. 1–7, doi: 10.1145/3229565.3229571.

[11] A. Hamza *et al.*, "Clear as MUD: Generating, validating and applying IoT behaviorial profiles," in *Proc. Workshop IoT Security Privacy*, 2018, pp. 8–14.

[12] Garcia *et al.*, "Enforcing behavioral profiles through software-defined networks in the industrial IoT," *App. Sci.*, vol. 9, 2019, Art. no. 2019.

[13] J. Voas *et al.*, NISTIR 8222: (IoT) Trust Concerns, NIST, 2018.

[14] Ma Zhaofeng, W. Xiaochang, D.K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2013–2021, Mar. 2020.

[15] K. Kataoka, S. Gangwar, and P. Podili, "Trust list: Internet-wide and distributed IoT traffic management using blockchain and SDN," in *Proc. IEEE World Forum Internet Things*, 2018, pp. 296–301.

[16] Y. Gao, X. Hu, H. Lin, Y. Liu, and L. Nie, "Blockchain based IIoT data sharing framework for SDN-enabled pervasive edge computing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5041–5049, Jul. 2021.

[17] D. Vishwasrao Medhane, A.K. Sangaiah, M. Shamim Hossain, G. Muhammad, and J. Wang, "Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6143–6149, Jul. 2020.

[18] A. Pillai, M. Sindhu, and K.V. Lakshmy, "Securing firmware in Internet of Things using blockchain," in *Proc. IEEE Int. Conf. Adv. Comput. Commun. Syst.*, 2019, pp. 329–334.

[19] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan "Smart contract-based access control for the internet of things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, Apr. 2019.

[20] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, and Y. Zhao, "EdgeChain: An edge-IoT framework and prototype based on blockchain and smart contracts," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4719–4732, Jun. 2019.

[21] X. Zhu, *Building a Secure Infrastructure for IoT Systems in Distributed Environments*. Lyon, France: Université de Lyon, NNT: 2019LYSEI038

[22] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[23] N. Koroniotis *et al.*, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019.

[24] Sharafaldin, CICIDS2017, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116

[25] Kitnet dataset, 2018. Online. Available: https://goo.gl/iShM7E

[26] P. Krishnan *et al.*, "SDN framework for securing IoT networks, Springer, Cham, Switzerland, vol. 218, 2018. Online. Available: https://doi.org/10.1007/978-3-319-73423-1_11