Received: 13 September 2023 | Revised: 30 October 2023 | Accepted: 24 November 2023

DOI: 10.1002/spy2.362

RESEARCH ARTICLE

WILEY

Designing access control security protocol for Industry 4.0 using Blockchain-as-a-Service

Anusha Vangala[1] | **Ashok Kumar Das**[1] | **Neeraj Kumar**[2] | **Pandi Vijayakumar**[3] | **Marimuthu Karuppiah**[4] | **Youngho Park**[5]

[1]Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India

[2]Department of Computer Science and Engineering, Thapar University, Patiala, India

[3]Department of Computer Science and Engineering, University College of Engineering Tindivanam, Villupuram, India

[4]School of Computer Science and Engineering & Information Science, Presidency University, Bengaluru, Karnataka, India

[5]School of Electronics Engineering, Kyungpook National University, Daegu, Republic of Korea

Correspondence
Ashok Kumar Das, Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, 500 032, India.
Email: iitkgp.akdas@gmail.com; ashok.das@iiit.ac.in

Funding information
Ministry of Electronics & Information Technology, New Delhi, Government of India, Grant/Award Number: 4(4)/2021-ITEA; Ripple Centre of Excellence (CoE) Scheme, CoE in Blockchain, IIIT Hyderabad, India, Grant/Award Number: IIIT/R&D Office/Internal Projects/001/2019; National Research Foundation of Korea, Grant/Award Number: 2020R1I1A3058605

Abstract

Industry 4.0 is a revolution of the operations in the industrial manufacturing for increased productivity, trade and commerce. It is heavily reliant on the automation of the processes and equipment along with complex interconnectivity and insightful analysis using machine learning. The interconnectivity of the manufacturing devices from various industrial sites brings with it several security issues related to communication. This article focuses on solving the security issue of access control between such devices and enable seamless secure communication for the proper functioning of the industry. An access control scheme has been proposed that achieves the necessary security features of anonymity, traceability, and forward secrecy. It is also shown that the proposed scheme takes less communication and computational costs, and is strongly resilient against various attacks such as impersonation attack, replay attack, and denial-of-service attack as compared to other relevant schemes.

KEYWORDS

access control, blockchain, Industrial Internet of Things (IIoT), Industry 4.0, security

1 | INTRODUCTION

The fourth industrial revolution, known as Industry 4.0 or Industrial Internet of Things (IIoT), is the application of the smart technology to the industrial goods manufacturing process used in factories. The production, assembly, repair, and maintenance of goods can be performed by leveraging the enabling technologies of Internet of Things (IoT), blockchain technology, robotic process automation (RPA), artificial intelligence (AI), big data, cloud computing, fog computing, edge computing, cyber physical systems, cognitive computing, and augmented reality. Such a digital transformation of factories propels partial transfer of autonomy and autonomous decisions to cyber-physical systems that leverage information systems to outsource work to other factories. This leads to a seamless integration of resources and increases the overall productivity of manufacturing processes leading to smart development of smart products. Such an amalgamation of communication of resources between industries necessitates to confront the security hazards of message exchange.[1,2] Blockchain technology has a strong opportunity to provide security to Industry 4.0 applications due to strong correlation in features of using digital identities, distributed security, micro-controls, and information transparency.[3,4]

1.1 | Network model

The network model shown in Figure 1 contains the hierarchy among the various nodes deployed in a cloud-based IoT-enabled smart manufacturing and production for Industry 4.0. At the smart factory level, we have various deployed IoT smart devices which communicate with their gateway node. The gateway node(s) are then connected to the respective fog server(s), which do the local processing of the data aggregated from the gateway nodes. The fog servers then provide the data to the cloud servers which form a blockchain network. Finally, the blockchain data is used for Big data analytics.

The network model consists of several smart devices equipped with smart sensors that are required during the manufacturing and production in the smart factories. The Industry 4.0 architecture has the capability to allow interoperability among the various smart factories by allowing device equipment nodes (DN) from a smart factory to access the DNs from other smart factory to outsource any overload of production work. Also, the DNs send their data to the gateway node GN associated with the smart factory. These scenarios require reliable access control that ensures the accessing user/device can only work with the required accessed device securely. The gateway node forwards the received device data to a fog server in the fog computing layer securely using the fog server's public key. This data is further forwarded to a cloud server in the Blockchain Cloud Center (BCC) securely via the cloud server's public key. The cloud servers perform validation and consensus of the block before adding it to the blockchain. The data from the blockchain can be requested by the Big Data Analytics Center (BDAC) in order to perform any analytics and processing on the data and obtain meaningful information reports that can help in proper decision making.

The cloud center consists of cloud servers which form a peer-to-peer network. The blockchain application is a distributed ledger technology that requires resources such as servers, nodes, databases, network setup and maintenance for the decentralized applications to run efficiently. These underlying resources for blockchain are provided by the peer-to-peer network of cloud servers. Thus, the cloud technology and blockchain technology complement each other. The purpose of fog computing layer is to balance the blockchain responsibilities on the cloud center and bring part of block creation closer to end nodes. So, the fog servers create partial blocks and send them to the cloud network. The cloud servers then perform the mining and consensus after creating the full blocks.

1.2 | Threat model

The adversary models that will be considered applicable for the proposed scheme are: (1) Dolev-Yao (DY) model,[5] (2) Wang model,[6] and (3) Canetti-Krawczyk (CK) adversary model.[7] The DY model assumes the adversary A has the capability to view, capture, modify, delete, and fabricate messages flowing in a communication channel. The adversary A can also impersonate one of the honest parties involved in the communication exchange. These capabilities give the adversary complete control over the network. The Wang adversary model is built over the DY model in order to add the capability to enumerate the identities and passwords that can be applied to the user within polynomial time. The adversary may either extract parameters from the smart card or gain the password. In addition, the adversary can extract all the established keys of the previous sessions and does not need clock synchronization across all entities in the network. The Wang adversary can also break two factors out of three and compromise sensor nodes to access their data. It can also obtain the secret key of gateway node and sensor nodes to obstruct forward secrecy. The fog server and cloud server are assumed to be trusted

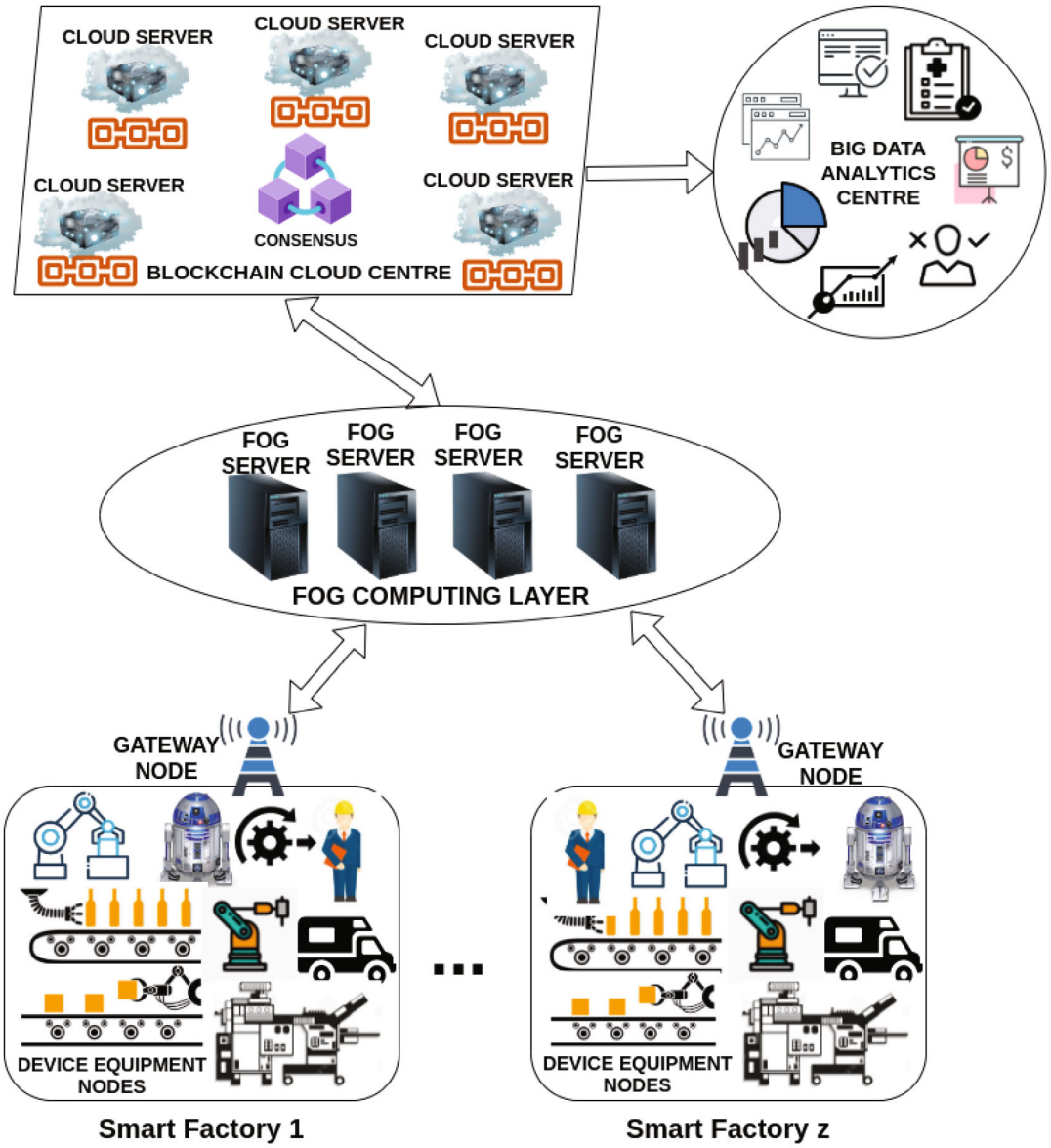

FIGURE 1 Cloud-based IoT-enabled smart manufacturing and production system for Industry 4.0.

and their secret keys are stored in their respective secure databases. In addition, the adversary A can impersonate as the gateway or sensor node. The CK-adversary model assumes that the adversary A can capture the long-term and short-term secrets along with the session states using session hijacking attack. It also assumes that the communicating end-point entities are untrustworthy. The smart devices in the network are assumed to be vulnerable to physical capturing by the adversary A to launch power analysis attacks,[8] which allows further attacks such as impersonation attacks. The servers used in the fog layer are assumed to use secure database storage that resists the stolen verifier attack and does not allow the attacker to have access to the credentials stored and prevents further attacks such as server impersonation attack.

1.3 | Research contributions

The following are the contributions of this article:

1. A novel blockchain-based access control scheme is designed for smart manufacturing environment with two phases: (a) device access control (DAC) between two device equipment nodes, and (b) gateway access control (GAC) between a device equipment node and a gateway node.

2. A novel architecture for Industry 4.0 based on fog computing, cloud computing and blockchain technology is proposed.
3. Blocks are formed from the encrypted sensor data received from the device equipment nodes and added to the blockchain using the "Practical Byzantine Fault Tolerance" consensus algorithm.
4. A detailed security analysis is performed using formal analysis and informal analysis. In addition, comparative study with relevant schemes shows that the proposed scheme can resist various attacks with minimal cost.

1.4 | Article outline

The article proposes a novel architecture for Industry 4.0 based on fog computing, cloud computing, and blockchain technology in Section 1.1. Section 2 studies various relevant schemes. The working of the proposed scheme is laid in Section 3. Section 4 gives a rigorous analysis of the proposed scheme. Section 5 provides a comparative analysis of the proposed scheme with related schemes. In Section 7, some open research problems have been discussed. Finally, Section 8 concludes the research work.

2 | RELATED WORK

This section studies the recent work done in authentication and key agreement in Industry 4.0.

Baruah and Dhal[9] propose an simple scheme for IIoT environment involving authenticating a sensor with a router in the IIoT network with the aid of an authentication server (AS) based on only hashing, XOR and exponentiation operations. Considering that their scheme does not specify an adversary model, the session key established consists of only short term random secrets and hence is vulnerable to "Ephemeral Secret Leakage (ESL) attack under CK adversary model." Compromise of the random session secret can also expose the identity of the sensor and lead to loss of anonymity.

Wang et al.[10] emphasize the importance of achieving perfect forward secrecy in the Industry 4.0 scenario where the increase in the number of sensor nodes and remote users overloads a gateway's computational and storage capabilities. They also show that most existing protocols are vulnerable to offline dictionary attacks. A protocol based on RSA with a small prime has been proposed to achieve forward secrecy in inexpensive computations. Using a small prime reduces the security of a scheme using RSA even if the operations are efficient, because the Fermat's factoring into small prime factors can be easily applied to break the scheme.

Zagrouba et al.[11] propose a "Blockchain Authentication and Trust Management (BATM)" consisting of encryption to achieve confidentiality, digital signatures to achieve authentication and peer identity validation to achieve trust. A single master key generates secondary keys that are used in encryption and digital signatures, which are renewed after a timeout. Each block to be added in the blockchain should be from an authenticated node with the payload content of network node condition and cryptographic data decided by the miner. The miner solves a difficult problem and approves the payload content before adding the block to the blockchain. The trust module surveils the node payload and gathers their credibility to generate reputation level and the trust level is computed from the number of authenticated nodes. However, this scheme lacks timely availability of services from IoT devices. The authors in References 12 and 13 provide a comprehensive literature survey on the usage of blockchain technology in Industry 4.0.

Esfahani et al.[14] study the various machine-to-machine (M2M) communication protocols applicable in IIoT and propose a scheme for M2M communication in an IIoT network where a machine consisting of a smart sensor is authenticated by a network element that uses a OPTIGA Trusted Platform Module (TPM) with crypto-controller security chips to achieve authentication with key management, certificate and password management. The sensor is first registered with an AS, which authenticates it with the router using hashing and XOR operations. Anonymity of the sensor is preserved by hiding its real identity. It is vulnerable to single point of failure, and stolen verifier, session key disclosure, and privileged insider attacks.

Garg et al.[15] proposed an authentication protocol for Industry 4.0 using "elliptic curve cryptography (ECC)," "physically unclonable functions (PUFs)," "hashing," and "bitwise XOR operations." An IoT node equipped with PUF also uses challenge-response pairs (CRPs) to authenticate with a remote server. The clocks of all the entities need to synchronized for the scheme to work. It is vulnerable to physical attack on devices as the scheme heavily relies on PUFs, in addition to ESL, and it does not also support anonymity and untraceability.

Kolluru et al.[16] propose an "Authentication, Authorization and Accounting system (AAA)" over a "Service Oriented Architecture (SOA) based Arrowhead framework" that is integrated with "Next Generation Access Control (NGAC)."

The Arrowhead software framework is hosted on the hardware IoT devices to provide interoperability and late run-time binding among loosely couples services. The AAA system provides a coarse grained access control at system level to consume all services from a service producer. The NGAC standard is added to provide attribute based fine grained access control among the basic elements of users, objects and operations, which can grouped using containers with the relations assignment, association, prohibition and obligation. All these are stored in policy information point (PIP). The access requests from users are intercepted using policy enforcement point (PEP). Authorization decisions are evaluated and issued using policy decision point (PDP). However, their scheme is based on Datagram Transport Layer Security (DTLS) using X.509 certificates, which creates a performance bottleneck for validation of public keys on resource-limited IoT devices.

Shuai et al.[17] proposed an authentication scheme for Industrial Internet of Things (IIoT) based on the Rabin cryptosystem. However, it is vulnerable to "user impersonation, stolen smart card, ESL, privileged insider and offline guessing attacks." Gupta et al.[18] proposed an identity-based authentication scheme between two users in different IoT regions. Li et al.[19] proved that the scheme in Reference 18 is vulnerable to ESL attack leading to impersonation attacks and insecure identity authentication. Rangwani et al.[20] proposed an ECC-based user authentication scheme that does not support three-factor authentication, and is vulnerable to gateway impersonation and replay attacks.

Tange et al.[21] surveyed the existing schemes to find the various security goals and how the fog computing can achieve the security requirements of IIoT. Yang et al.[22] proposed a protocol using symmetric cryptography and hash functions. However, it lacks perfect forward secrecy since the public key cryptography (PKC) is not used. Srinivas et al.[23] proposed an authentication scheme using the chaotic maps based on the Chebyshev polynomial. Vinoth et al.[24] proposed a lightweight scheme based on secret sharing and access structures.

Abdi Nasib Far et al.[25] identified that Vinoth et al.'s scheme[24] is vulnerable to replay attack, "Denial-of-Service (DoS)" attack, de-synchronization attack and sensor node capture attack. Raque et al.[26] proposed a scheme using hash functions and symmetric encryption/decryption with de-synchronization of pseudo-identities. Karati et al.[27] presented a certificate-less scheme that provided data integrity in IIoT. However, Rezaeibagha et al.[28] and Zhang et al.[29] proved that the scheme of Karati et al.[27] is vulnerable to signature forgery attacks.

We now look at symmetric key based authentication protocols in IoT. Avoine et al.[30] proposed "Symmetric-key Authenticated Key Exchange (SAKE)" and SAKE-AM, which were found to be susceptible to time-based attacks for concurrent sessions, replay attacks, and inability to provide untraceability by Aghili et al.,[31] who proposed the scheme called as "Strengthened Symmetric-Key Authenticated Key Exchange" (SAKE+). Ferreira[32] found that SAKE+ is still susceptible to replay, timing, tracking and disconnection attacks, while breaking the entity authentication property. Fan et al.[33] proposed SAKE* that claims to be secure under the Brzuka threat model, but it does not support anonymity, untraceability, ESL and Denial of Service (DoS) attacks.

It can be understood from the above discussion that the existing schemes either have a performance bottleneck or are vulnerable to various attacks or lack required security functionalities necessary for an access control scheme to be applied for Industry 4.0. Hence, it paves the way for the design of a more comprehensive access control protocol within limited computational and communicational costs.

3 | THE PROPOSED SCHEME

In this section, we give various phases associated with the proposed blockchain-enabled access control scheme for Industry 4.0 environment. In a smart factory, the equipments used are registered with one smart factory and rarely moved/roamed to another factory. Hence, during the registration phase itself, the required credentials of each device are loaded into the secure memory of these devices. Note that the mutual authentication between devices and gateway nodes helps for secure communication between them.

The novelty behind the proposed scheme is as follows. The authentication scheme proposed in this work uses the random nonces of both the device and gateway that contribute to the session key established in the authentication and key agreement scheme along with the long term secret credentials so that it becomes resistance to the "ESL" attack under the CK-adversary model (described in Section 4.2.6). The proposed scheme allows the secure data aggregation before block creation, verification, and addition in the blockchain so that only the genuine IoT smart devices data is inserted into the blockchain. Moreover, the proposed scheme allows Big data analytics on the blockchain data to be used for consumer consumption, decision-making, optimization, targeted marketing, and development of products. In addition, the proposed also offers privacy preservation to ensure that if any intermediate parties involved in communication cannot

TABLE 1 Notations and their description.

Notation	Significance
$E_q(a, b)$	A non-singular elliptic curve of the form: $y^2 = x^3 + ax + b \ (mod\ q)$
P	A base point in $E_q(a, b)$ of order n_P as big as q
$x \cdot P$	An elliptic curve point multiplication: $x \cdot P = P + P + \cdots + P$ (x times)
$P + Q$	Elliptic curve point addition; $P, Q \in E_q(a, b)$
RA	Registration authority
GN_k	Gateway node
DN_i	IoT smart (sensor) device
FS_l	Fog server
CS_y	Cloud server
RTS_X	Registration timestamp issued by the RA to an entity X
ID_X, TID_X, RID_X	Entity X's real, temporary and pseudo identities, respectively
ck_{RA}	Master secret key of RA
pr_X, Pub_X	Private and public keys of entity X, respectively
n_i, r_i, r_2, e_1, c_1	RA's random secrets
d_i, r_i, s_i	DN_i's random secrets
g_k, w_k	GN_k's random secrets
f_l	FS_l's random secrets
h_y	CS_y's random secrets
$\|\|$	Concatenation operation
$T_i, T_j, T_k, TSP_1, TSP_2, TSP_3$	Current timestamps
$*$	Modular multiplication in a finite field Z_q
$\oplus$	Exclusive-OR (XOR) operation
ΔT	Maximum transmission delay for a message
$H(\cdot)$	"Collision-resistant cryptographic one-way hash function"

know what data is being communicated. The secret credentials used in the registration and authentication phases are the critical and these are never shared in any public channel directly.

The list of notations and their descriptions is tabulated in Table 1.

3.1 | System initialization phase

In this phase, a trusted registration authority (*RA*) is responsible for picking the system parameters with the help of the following steps:

Step SIP_1: The RA picks a "non-singular elliptic curve of the form: $E_q(a, b) : y^2 = x^3 + ax + b \ (mod\ q)$ over the Galois field $GF(q)$," with a "point at infinity (zero point) $\mathcal{O}$, constants $a, b \in Z_q = \{0, 1, 2, \ldots, q-1\}$ such that the condition $4a^3 + 27b^2 \neq 0 \ (mod\ q)$ is satisfied." The RA then includes a "base point $G \in E_q(a, b)$ with an order n_G as large as q," implying that "$n_G \cdot G = G + G + \cdots + G$ (n_G times) $= \mathcal{O}$".[34]

Step SIP_2: The RA picks a "collision-resistant one-way cryptographic hash function, say $H(\cdot)$ (for instance, SHA-256 hash algorithm may be used[35])" and the "Practical Byzantine Fault Tolerance (PBFT)" algorithm for achieving consensus in the BCC.

Step SIP_3: Finally, the RA chooses its "master secret key as ck_{RA} in Z_q^*" and then publishes the "domain parameters $\{E_q(a, b), G, H(\cdot)\}$ as public."

3.2 | Registration phase

This phase involves the registration of various entities, like IoT smart device (DN_i), gateway node (GN_k), fog server (FS_l) and cloud server (CS_y).

3.2.1 | Device node registration phase

This phase registers a device equipment node DN_i from a smart factory with the central registration authority RA.

1. *Step* DNR_1: RA picks its true identity ID_{N_i}, registration random number n_i, device random secret $d_i \in Z_q^*$, device registration timestamp RTS_{N_i}, and computes the pseudo-identity for the device equipment node as $RID_{N_i} = H(ID_{N_i}||n_i||RTS_{N_i}||ck_{RA})$.
2. *Step* DNR_2: RA picks the private key for the sensor device node as $pr_{N_i} \in Z_q^*$ and the corresponding public key as $Pub_{N_i} = pr_{N_i} \cdot P$.
3. *Step* DNR_3: RA preloads DN_i with $\{RID_{N_i}, d_i, H(\cdot), E_q(a,b), P, (pr_{N_i}, Pub_{N_i})\}$. It is worth noticing that the device random secret d_i is a long term secret generated by the central registration authority RA for the device equipment node and aids in protecting against ESL attack under the CK-adversary model.

3.2.2 | Gateway node registration phase

This phase registers a gateway node GN_k from a smart factory with the central registration authority RA.

1. *Step* GNR_1: GN_k picks its true identity ID_{G_k} and forwards it to RA via a secure channel. RA picks a registration random number $m_k \in Z_q^*$, gateway random secret $g_k \in Z_q^*$, gateway registration timestamp RTS_{G_k}, and computes the pseudo-identity for the gateway as $RID_{G_k} = H(ID_{G_k}||m_k||RTS_{G_k}||ck_{RA})$. RA sends RID_{G_k} and g_k to GN_k via secure channel.
2. *Step* GNR_2: GN_k picks the private key for the gateway node as $pr_{G_k} \in Z_q^*$ and the corresponding public key as $Pub_{G_k} = pr_{G_k} \cdot P$ to be used in computations in asymmetric cryptography.
3. *Step* GNR_3: GN_k stores $\{RID_{G_k}, g_k, H(\cdot), E_q(a,b), P, (pr_{G_k}, Pub_{G_k})\}$ in its secure memory storage. The gateway random secret g_k is a long term secret generated by the central registration authority RA for the gateway node and aids in protecting against ESL attack.

3.2.3 | Fog server registration phase

This phase registers a fog server FS_l from the fog computing layer with the central registration authority RA.

1. *Step* FSR_1: FS_l picks ID_{F_l} and forwards it to RA via a secure channel. RA picks a registration random number $f_l \in Z_q^*$ and a fog server registration timestamp RTS_{F_l} to compute the pseudo-identity for the fog server as $RID_{F_l} = H(ID_{F_l}||f_l||RTS_{F_l}||ck_{RA})$. RA sends RID_{F_l} to FS_l via secure channel.
2. *Step* FSR_2: FS_l picks the private key for the fog server as $pr_{F_l} \in Z_q^*$ and the corresponding public key as $Pub_{F_l} = pr_{F_l} \cdot P$.
3. *Step* FSR_3: FS_l stores $\{RID_{F_l}, H(\cdot), E_q(a,b), P, (pr_{F_l}, Pub_{F_l})\}$ in its secure memory storage.

3.2.4 | Cloud server registration phase

This phase registers a cloud server CS_y from the BCC with the central registration authority RA.

1. *Step* CSR_1: CS_y picks its true identity ID_{CS_y} and forwards it to RA via a secure channel. RA picks a registration random number $h_y \in Z_q^*$ and a cloud server registration timestamp RTS_{CS_y} in order to compute the pseudo-identity for the cloud server as $RID_{CS_y} = H(ID_{CS_y}||h_y||RTS_{CS_y}||ck_{RA})$. RA sends RID_{CS_y} to CS_y via secure channel.

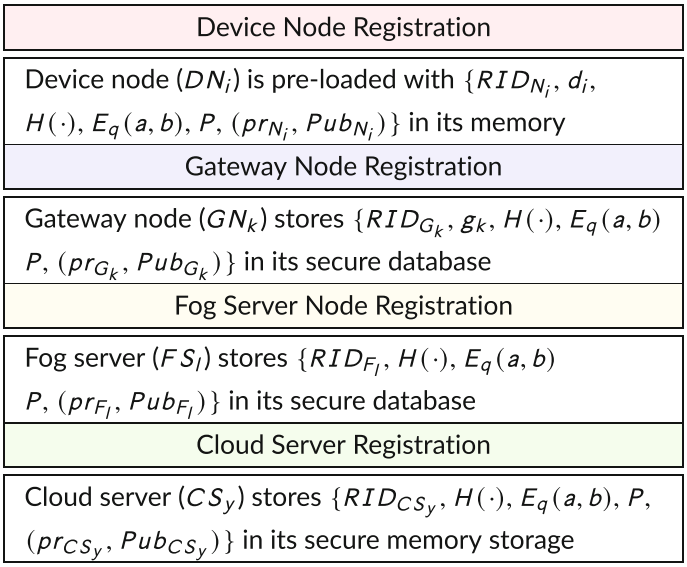

FIGURE 2 Summary of registration phases.

2. *Step* CSR_2: CS_y picks the private key for the cloud server as $pr_{\text{CS}_y} \in Z_q^*$ and the corresponding public key as $\text{Pub}_{\text{CS}_y} = pr_{\text{CS}_y} \cdot P$.
3. *Step* CSR_3: CS_y stores $\{\text{RID}_{\text{CS}_y}, H(\cdot), E_q(a,b), P, (pr_{\text{CS}_y}, \text{Pub}_{\text{CS}_y})\}$ in its secure memory storage.

The important credentials stored during the registration phases for various devices, gateway node, fog server node and cloud server are summarized in Figure 2.

3.3 | Access control phase

In this section, we propose two types of access control mechanisms: (1) *DAC* in which two devices verify their credentials and establish a session key, and (2) *GAC* in which a device and the gateway verify credentials and establish a session key.

3.3.1 | Device access control phase

This phase allows a device equipment node DN_i from a smart factory to access the device equipment node DN_j from another smart factory. For such an access, the two devices verify their credentials and establish a session key using the following steps:

Step DAC_1: Device equipment node DN_i generates a random secret $r_i \in Z_q^*$ and a timestamp $T_i \in Z_q^*$. It generates a private $rd_i = (r_i + d_i)\ (mod\ q)$ using device random number d_i received in registration phase (see Section 3.2.1) and computes $\text{RD}_i = rd_i \cdot P$. To ensure anonymity of pseudo-identity RID_{N_i}, it computes a session identity $\text{RID}'_{N_i} = H(rd_i||\text{RID}_{N_i}||pr_{N_i}||T_i)$. Then a private hash $g_i = H(r_i||d_i||pr_{N_i}||T_i)$ is generated to be used in the session key. The public equivalent of g_i is computed as $G_i = g_i \cdot P$. A signature is computed as $\text{Sig}_i = rd_i + H(\text{RID}'_{N_i}||\text{Pub}_{N_i}||G_i||\text{RD}_i||T_i) * pr_{N_i}\ (mod\ q)$. The message $\text{Msg}_{\text{DAC}_1} : \langle \text{RID}'_{N_i}, \text{RD}_i, G_i, T_i, \text{Sig}_i \rangle$ is sent to the device equipment node DN_j which is to be accessed by DN_i.

Step DAC_2: DN_j receives $\text{Msg}_{\text{DAC}_1}$ at time T_j and verifies the timestamp $|T_j - T_i| \leq \Delta\text{T}$, and signature $\text{Sig}_i \cdot P \stackrel{?}{=} \text{RD}_i + H(\text{RID}'_{N_i}||\text{Pub}_{N_i}||G_i||\text{RD}_i||T_i) \cdot \text{Pub}_{N_i}$. If verified to be correct, it generates a random secret $r_j \in Z_q^*$, computes a private $rd_j = (r_j + d_j)\ (mod\ q)$ using device random number d_j received in registration phase (see Section 3.2.1) and its public equivalent as $\text{RD}_j = rd_j \cdot P$. It computes its session identity $\text{RID}'_{N_j} = H(rd_j||\text{RID}_{N_j}||pr_{N_j}||T_j)$ to hide RID_{N_j}.

Smart Device Equipment Node (DN_i)	Smart Device Equipment Node (DN_j)
Generate $r_i, T_i \in Z_q^*$	
Compute $rd_i = (r_i + d_i) \pmod q$,	
$RD_i = rd_i \cdot P$,	
$RID'_{N_i} = H(rd_i \| RID_{N_i} \| pr_{N_i} \| T_i)$,	
$g_i = H(r_i \| d_i \| pr_{N_i} \| T_i)$,	
$G_i = g_i \cdot P$,	Check if $\|T_j - T_i\| \leq \Delta T$?
$Sig_i = rd_i + H(RID'_{N_i} \| Pub_{N_i} \| G_i \|$	If so, verify $Sig_i \cdot P \stackrel{?}{=} RD_i + H(RID'_{N_i} \|$
$RD_i \| T_i) * pr_{N_i} \pmod q$	$Pub_{N_i} \| G_i \| RD_i \| T_i) \cdot Pub_{N_i}$
$Msg_{DAC_1} : \langle RID'_{N_i}, RD_i, G_i, T_i, Sig_i \rangle$	If yes, generate r_j
$\longrightarrow$	
	Compute $rd_j = (r_j + d_j) \pmod q$,
	$RD_j = rd_j \cdot P$,
	$RID'_{N_j} = H(rd_j \| RID_{N_j} \| pr_{N_j} \| T_j)$,
	$g_j = H(r_j \| d_j \| pr_{N_j} \| T_j)$,
Check if $\|T_k - T_j\| \leq \Delta T$?	$G_j = g_j \cdot P$,
If so, compute	$SK_{ji} = H(g_j \cdot G_i \| RID'_{N_i} \| RID'_{N_j} \| T_i \| T_j)$,
$SK_{ij} = H(g_i \cdot G_j \| RID'_{N_i} \| RID'_{N_j} \| T_i \| T_j)$	$Sig_j = rd_j + H(RID'_{N_j} \| SK_{ji} \| Pub_{N_j} \|$
Verify if $Sig_j \cdot P \stackrel{?}{=} RD_j + H(RID'_{N_j} \|$	$G_j \| RD_j \| T_j) * pr_{N_j} \pmod q$
$SK_{ij} \| Pub_{N_j} \| G_j \| RD_j \| T_j) \cdot Pub_{N_j}$	$Msg_{DAC_2} : \langle RID'_{N_j}, RD_j, G_j, T_j, Sig_j \rangle$
	$\longleftarrow$
If yes, store SK_{ij}	Store SK_{ji}

FIGURE 3 Summary of device access control (DAC) phase.

Step DAC_3: DN_j then computes the private hash $g_j = H(r_j||d_j||pr_{N_j}||T_j)$ to be part of the session key along with its public equivalent $G_j = g_j \cdot P$. The session key is computed as $SK_{ji} = H(g_j \cdot G_i||RID'_{N_i}||RID'_{N_j}||T_i||T_j)$ along with the signature on private rd_j as $Sig_j = rd_j + H(RID'_{N_j}||SK_{ji}||Pub_{N_j}||G_j||RD_j||T_j) * pr_{N_j}$ $(mod\ q)$. DN_j replies DN_i with the message $Msg_{DAC_2} : \langle RID'_{N_j}, RD_j, G_j, T_j, Sig_j \rangle$. *Step* DAC_4: DN_j receives Msg_{DAC_2} at time T_k and checks if $|T_k - T_j| \leq \Delta T$. If so, It computes the session key as $SK_{ij} = H(g_i \cdot G_j||RID'_{N_i}||RID'_{N_j}||T_i||T_j)$ and verifies the received signature as $Sig_j \cdot P \stackrel{?}{=} RD_j + H(RID'_{N_j}||SK_{ij}||Pub_{N_j}||G_j||RD_j||T_j) \cdot Pub_{N_j}$. If the signature is verified, DN_i stores SK_{ij} and DN_j stores SK_{ji}.

The summary of the DAC phase is given in Figure 3.

3.3.2 | Gateway access control phase

This phase allows a device equipment node DN_i from a smart factory to access the gateway node GN_k. For such an access, first the device and the gateway verify credentials and establish a session key using the following steps:

Step GAC_1: Device equipment node DN_i generates a random secret $s_i \in Z_q^*$ and a timestamp $TSP_1 \in Z_q^*$. It generates a private $sd_i = (s_i + d_i)$ $(mod\ q)$ using device random number d_i received in registration phase (see Section 3.2.1) and computes $SD_i = sd_i \cdot P$. To ensure anonymity of pseudo-identity RID_{N_i}, it computes $RID'_{N_i} = H(sd_i||RID_{N_i}||pr_{N_i}||TSP_1)$. Then a private hash $u_i = H(s_i||d_i||pr_{N_i}||TSP_1)$ is generated to be used in the session key. The public equivalent of u_i is computed as $U_i = u_i \cdot P$. A signature is computed as $Sig_{D_i} = sd_i + H(RID'_{N_i}||Pub_{N_i}||U_i||SD_i||TSP_1) * pr_{N_i}$ $(mod\ q)$. The message $Msg_{GAC_1} : \langle RID'_{N_i}, SD_i, U_i, TSP_1, Sig_{D_i} \rangle$ is sent to the gateway node GN_k.

Step GAC_2: GN_k receives Msg_{GAC_1} at time TSP_2 and verifies the timestamp $|TSP_2 - TSP_1| \leq \Delta T$, and signature $Sig_{D_i} \cdot P \stackrel{?}{=} SD_i + H(RID'_{N_i}||Pub_{N_i}||U_i||SD_i||TSP_1) \cdot Pub_{N_i}$. If verified to be correct, it generates a random secret $w_k \in Z_q^*$, computes a private $wg_k = (w_k + g_k)$ $(mod\ q)$ using gateway random number g_k received in registration phase Section 3.2.2 and its public equivalent as $WG_k = wg_k \cdot P$. It computes $RID'_{G_k} = H(wg_k||RID_{G_k}||pr_{G_k}||TSP_2)$ to hide RID_{G_k}.

Smart Device Equipment Node (DN_i)	Gateway Node (GN_k)
Generate $s_i, TSP_1 \in Z_q^*$	
Compute $sd_i = (s_i + d_i) \pmod q$,	
$SD_i = sd_i \cdot P$,	
$RID'_{N_i} = H(sd_i \| RID_{N_i} \| pr_{N_i} \| TSP_1)$,	Check if $\|TSP_2 - TSP_1\| \leq \Delta T$?
$u_i = H(s_i \| d_i \| pr_{N_i} \| TSP_1)$,	If so, verify $Sig_{D_i} \cdot P \stackrel{?}{=} SD_i + H(RID'_{N_i} \| Pub_{N_i} \| U_i \| SD_i \| TSP_1) \cdot Pub_{N_i}$
$U_i = u_i \cdot P$,	If yes, generate w_k
$Sig_{D_i} = sd_i + H(RID'_{N_i} \| Pub_{N_i} \| U_i \| SD_i \| TSP_1) * pr_{N_i} \pmod q$	Compute $wg_k = (w_k + g_k) \pmod q$,
$Msg_{GAC_1} : \langle RID'_{N_i}, SD_i, U_i, TSP_1, Sig_{D_i} \rangle \longrightarrow$	$WG_k = wg_k \cdot P$,
	$RID'_{G_k} = H(wg_k \| RID_{G_k} \| pr_{G_k} \| TSP_2)$,
	$v_k = H(w_k \| g_k \| pr_{G_k} \| TSP_2)$,
	$V_K = v_k \cdot P$,
	$SK_{ki} = H(v_k \cdot U_i \| RID'_{N_i} \| RID'_{G_k} \| TSP_1 \| TSP_2)$
	$Sig_{G_k} = wg_k + H(RID'_{G_k} \| SK_{ki} \| Pub_{G_k} \| V_k \| WG_k \| TSP_2) * pr_{G_k} \pmod q$
Check if $\|TSP_3 - TSP_2\| \leq \Delta T$?	$Msg_{GAC_2} : \langle RID'_{G_k}, WG_k, V_k, TSP_2, Sig_{G_k} \rangle \longleftarrow$
If so, compute $SK_{ik} = H(u_i \cdot V_k \| RID'_{N_i} \| RID'_{G_k} \| TSP_1 \| TSP_2)$	
Verify $Sig_{G_k} \cdot P \stackrel{?}{=} WG_k + H(RID'_{G_k} \| SK_{ik} \| Pub_{G_k} \| V_k \| WG_k \| TSP_2) \cdot Pub_{G_k}$	
If yes, store SK_{ik}	Store SK_{ki}

FIGURE 4 Summary of gateway access control (GAC) phase.

Step GAC_3: GN_K then computes the private hash $v_k = H(w_k||g_k||pr_{G_k}||\text{TSP}_2)$ to be part of the session key along with its public equivalent $V_k = v_k \cdot P$. The session key is computed as $\text{SK}_{ki} = H(v_k \cdot U_i||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2)$ along with the signature on private wg_k as $\text{Sig}_{G_k} = wg_k + H(\text{RID}'_{G_k}||\text{SK}_{ki}||\text{Pub}_{G_k}||V_k||\text{WG}_k||\text{TSP}_2) * pr_{G_k}\ (mod\ q)$. GN_k replies DN_i with the message $\text{Msg}_{\text{GAC}_2} : \langle \text{RID}'_{G_k}, \text{WG}_k, V_k, \text{TSP}_2, \text{Sig}_{G_k} \rangle$.

Step GAC_4: DN_i receives $\text{Msg}_{\text{GAC}_2}$ at time TSP_3 and checks if $|\text{TSP}_3 - \text{TSP}_2| \leq \Delta T$. If so, It computes the session key as $\text{SK}_{ik} = H(u_i \cdot V_k||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2)$ and verifies the received signature as $\text{Sig}_{G_k} \cdot P \stackrel{?}{=} \text{WG}_k + H(\text{RID}'_{G_k}||\text{SK}_{ik}||\text{Pub}_{G_k}||V_k||\text{WG}_k||\text{TSP}_2) \cdot \text{Pub}_{G_k}$. If the signature is verified, DN_i stores SK_{ik} and GN_k stores SK_{ki}.

The summary of GAC phase is also given in Figure 4.

3.4 | Dynamic node addition and device node revocation phase

Increase in the number of inputs occurs when the device equipment nodes increase across various smart factories. To accommodate increase in the device equipment nodes, a dynamic node addition phase has been proposed in this section. The proposed scheme supports addition of any number of new devices that can collect sensor inputs and send them to the BCC via the gateway node and fog computing layer.

If a device equipment nodes fails due to physical, environmental, or any other reason, it needs to be replaced with a new sensor device. Also, there may be cases where a device in working condition may have to be revoked due to security, legal or other reasons. In such cases, a new device needs to be registered using the procedure in Section 3.2.1 and we need to replace the old device.

3.5 | Secure data aggregation with block creation, verification and addition in BC

This phase is concerned with assuring that the data from the device equipment nodes are converted into transactions, which are verified and stored onto the blockchain. It involves the roles of the gateway node, fog server and multiple cloud servers from the architecture shown in Figure 1.

Block Header	
Block Version ($BlockVer$)	Unique block version number
Previous Block Hash ($Prev_BHash$)	Hash value of previous block
Merkle Tree Root ($MTR_{ParBlock_l}$)	Merkle tree root on transactions
Timestamp ($Time_{ParBlock_l}$)	Block creation time
Owner of Block	Fog Server FS_l
Public key of transactions verification	Pub_{F_l}
Block Payload (Encrypted Transactions)	
Encrypted Transactions Tx_1	$(Enc_{Pub_{F_l}}(TX_k), Enc_{Pub_{F_l}}(Sign_{TX_k}))$ ⋮
ECDSA signature on Partial Block	$Sign_{ParBlock}$
Current Block Hash ($Curr_BHash$)	Hash value of current block

FIGURE 5 Structure of a full block $FullBlock_y$ in the blockchain.

An industrial factory manufactures parts of a product in various stages. While manufacturing a part, it may require the data regarding other parts that form the product together. The gateway node and the fog server receive the data from the device equipment nodes in encrypted format. But neither of these are allowed to store such sensitive data in their memory. Due to this, when previous data from device equipment nodes are required, the gateway and fog servers cannot obtain this data. Such sensitive data is only stored in the blockchain ledger with immutability and transparency. Therefore, using blockchain reduces the burden of secure data storage from the gateway nodes and fog servers.

Step DABC$_1$: The gateway node GN_k collects all the data encrypted using the session key established in the GAC phase as shown in Section 3.3.2 from the device equipment nodes in the smart factories under it. The GN_k then generates transactions $TX_k = (RID_{G_k}, TS_{GN}, DNDATA_i)$ and signs it using the "elliptic curve digital signature algorithm (ECDSA)" denoted as $Sign(\cdot)$ as $Sign_{TX_k} = Sign_{pr_{G_k}}(H(RID_{G_k}||TS_{GN}||DNDATA_i))$ where TS_{GN} is the current timestamp. GN_k then sends the message $\langle Enc_{Pub_{F_l}}(TX_k), Enc_{Pub_{F_l}}(Sign_{TX_k})\rangle$ to the associated fog server FS_l after encrypting using its public key Pub_{F_l}.

Step DABC$_2$: FS_l decrypts the received message using its private key pr_{F_l} and verifies the received transaction by validating the signature using Pub_{G_k} for the sending GN_k. It collects num_{tr} validated transactions from various gateway nodes and creates a partial block $ParBlock_l$ containing the block version $BlockVer$, block genesis timestamp $Time_{ParBlock_l}$, Merkle Tree Root $MTR_{ParBlock_l}$ on the num_{tr} transactions, block owner as FS_l, public key for transaction verification as Pub_{G_k}, and partial block signature over the transactions and block header as $Sign_{ParBlock}$ using its private key pr_{F_l}. $ParBlock_l$ is then sent to the associated cloud server CS_y.

*Step B*DAC$_3$: CS_y decrypts the received transaction using pr_{CS_y} and verifies $Sign_{ParBlock}$ using Pub_{F_l} given in $ParBlock_l$. If successfully verified, CS_y adds the hash of the previous block in the blockchain $Prev_BHash$, computes the new hash of the current block as $Curr_BHash$ on the partial block with the newly added $Prev_BHash$, and generates the full block $FullBlock_y$. Figure 5 shows the structure of the full block.

*Step B*DAC$_4$: $FullBlock_y$ will be distributed to all the cloud servers that are allowed to participate the consensus process. One of the cloud servers in the BCC will be elected as the leader. The cloud servers will then execute the "Practical Byzantine Fault Tolerance ($PBFT$)" to achieve consensus on the newly received $FullBlock_y$. Considering $Count_{Faulty_CS}$ is the number of faulty cloud servers, if more than $2 * Count_{Faulty_CS}$ verify the block signatures, timestamps, Merkle tree root and block hash successfully, then the leader cloud server commands all the cloud servers to commit the block and add it their respective copies of the blockchain.

> *Remark.* Increase in the number of device equipment nodes increase the amount of sensor data. As this increases, the number of transactions in each block can be increased. This can keep the number of blocks in the blockchain in control. The value of num_{tr} at each fog server FS_l may be increased to increase the number

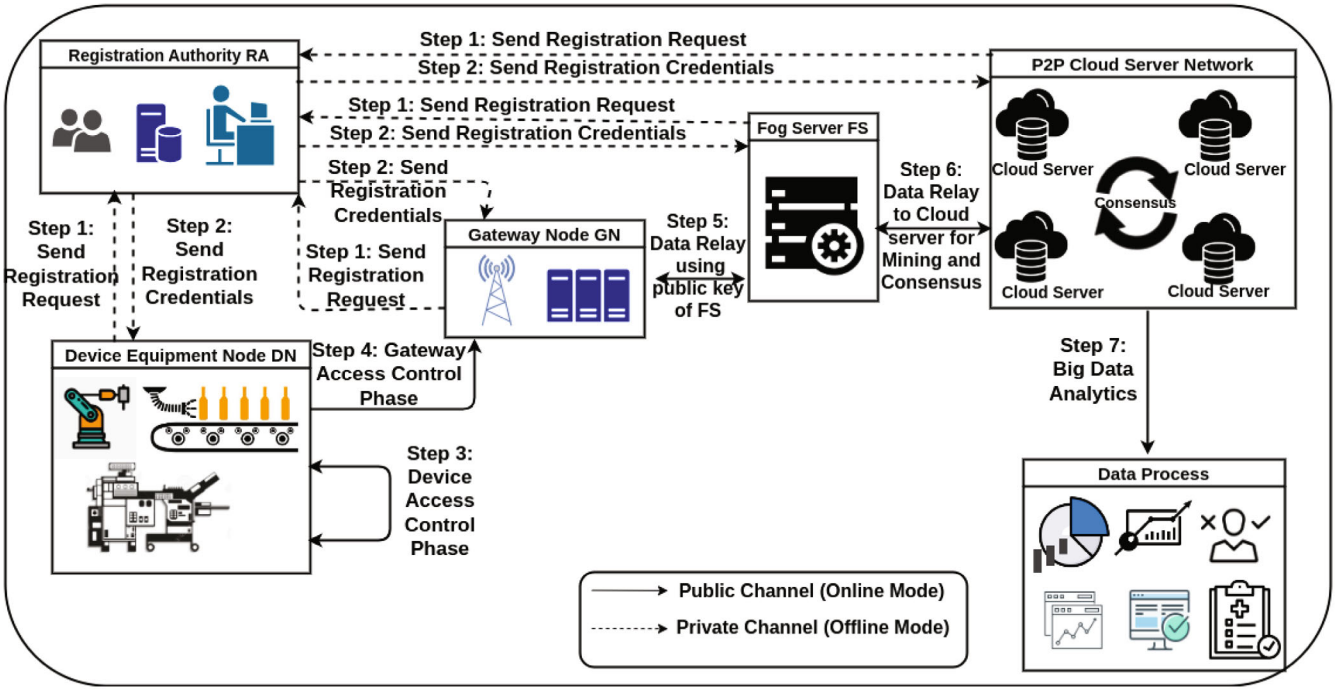

FIGURE 6 Process flow diagram for cloud-based IoT-enabled smart manufacturing and production system for Industry 4.0.

of transactions in each block. The scalability has been studied as a simulation study in Section 6 that shows only a linear increase in the computational time with increase in P2P nodes, blocks and transactions in each block.

Remark. Blockchain technologies require a stable core network of nodes. This requirement is achieved by using a number of cloud servers at the BCC of the proposed model. These cloud servers act as a stable set of core network nodes that are involved in performing the consensus, mining and block addition onto chain on the full blocks created using the partial blocks from the fog servers. The fog servers only create partial blocks from the collected transactions but do not add them to any blockchain.

3.6 | Big data analytics phase

The BDAC requests for the required data from the BCC as and when need arises as per the application using this data. When the required data is received, it is evaluated to see if it applies to the business case and the appropriate data is identified and extracted. This extracted data is organized and partitioned logically so that it is compatible for querying. This processed data is then checked critically for quality and any discovered errors or inconsistencies are rectified. The data may be summarized and aggregated. The clean aggregated data may then be analyzed thorough one or more techniques such as data mining, text mining, machine learning, deep learning, artificial intelligence, and data visualization. The results of such analysis are modeled and used for consumer consumption, decision-making, optimization, targeted marketing, and development of products.

The process flow of the proposed model has been shown in Figure 6. The flow starts with the registration authority RA registering the device equipment nodes DN, gateway node GN, the fog servers FS, and the cloud servers CS. The device nodes then execute the proposed DAC phase and then securely access data from one device to another. The gateway node then executes the GAC phase with the device nodes before securely accessing data from them. The data are then relayed securely from the gateway node to the fog servers in the form of transactions and from fog servers to the cloud servers as partial blocks as shown in the secure data aggregation phase discussed in Section 3.5. The cloud servers create the full blocks and add to the blockchain after consensus. This data is provided to BDAC to execute the Big data analytics phase as described in Section 3.6.

TABLE 2 Queries and their purposes.

Query	Purpose
$Execute(\varphi_{DN_1}^{x_1}, \varphi_{DN_2}^{x_2}, \varphi_{GN}^{x_3})$	The messages communicated among DN_1, DN_2, and GN are made available to $\mathcal{A}$
$CorruptSD(\varphi_{DN_i}^{x})$	$\mathcal{A}$ acquires the pre-saved secret credentials stored in a compromised smart device DN with this query
$Reveal(\varphi^{x})$	$\mathcal{A}$ acquires the session key between φ^{x} and its respective participant with the help of this query
$Test(\varphi^{x})$	The established session keys SK_{ij} (= SK_{ji}) between DN_1 and DN_2 and SK_{ij} (= SK_{ji}) between DN and GN are checked to determine if they are originally established keys or some random keys

Remark. The proposed approach is independent of the type of IIoT devices used in the smart factories. The scheme is designed to be effectively used for authenticating device equipment nodes before transmission of any data to the gateway node. Any device that is suitable in the Industry 4.0 scenario needs to be registered using the *device node registration phase* discussed in Section 3.2.1. Such a registered device then works in its deployed environment to collect data using smart sensors in the device. When the device needs to send its data, it participates in the *DAC phase* and *GAC phase* to authenticate the devices and gateway, and establish session keys to encrypt the data before sending.

4 | SECURITY ANALYSIS

In this section, we utilize the "formal and informal security analysis" to show that the proposed scheme is robust against various potential attacks.

4.1 | Formal security analysis under ROR model

In the proposed scheme *BACPI*4, the DAC phase establishes a session key between two device equipment nodes belonging to same or different factory and the GAC phase establishes another session key between a device equipment node and the gateway node for a smart factory. The ROR model specifies the security of the established session keys using Theorem 1 that is based on Definition 1. A "one-way cryptographic hash function" $H(\cdot)$ that is considered as a random oracle, *Hash* is available to the entities present in the access control scheme. In addition, adversary $\mathcal{A}$ utilizes the queries listed in Table 2.

Let $\varphi_{DN_1}^{x_1}$, $\varphi_{DN_2}^{x_1}$, and $\varphi_{GN}^{x_2}$ denote the x_1th, x_2th, and x_3th instances of DN_1, DN_2, and GN, respectively, which are known as "random oracles."

Definition 1. Consider that $Adv_{\mathcal{A}}^{BACPI4}(time_{poly})$ indicates the "advantage gained by an adversary $\mathcal{A}$, which executes in polynomial time $time_{poly}$ in breaking the semantic security of the proposed BACPI4 for deriving the session key SK_{ij} (= SK_{ji}) between two smart device equipment nodes DN_i and DN_j in the DAC phase and the session key SK_{ik} (= SK_{ki}) between a device node DN_i and its gateway node GN_k in the GAC phase of the proposed BACPI4 in a particular session." Then, $Adv_{\mathcal{A}}^{BACPI4}(time_{poly}) = |2Pr[bit' = bit] - 1|$, where *bit* and *bit'* denote the "correct" and "guessed" bits, respectively.

Theorem 1. The considered adversary $\mathcal{A}$ from the threat model in Section I-B is assumed to execute in polynomial time $time_{poly}$ that attempts to obtain the session key SK_{ij}(= SK_{ji}) between two smart device equipment nodes DN_i and DN_j in the DAC phase, and the session key SK_{ik} (= SK_{ki}) between a device node DN_i and its gateway node GN_k in the GAC phase for a particular session of the proposed BACPI4. Let q_h be the "number of *Hash* queries," $|Hash|$ be the "range space of a one-way collision-resistant hash function $H(\cdot)$" and $Adv_{\mathcal{A}}^{ECDDHP}(time_{poly})$ be the "advantage in breaking the Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP)," respectively. Then $Adv_{\mathcal{A}}^{BACPI4}(time_{poly}) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(time_{poly})$.

Proof. In the proposed *BACPI*4, a set of three games are executed in sequence by the adversary $\mathcal{A}$, say $Game_x^{\mathcal{A}}$, $(x = 0, 1, 2)$, where $Success_{Game_x}^{\mathcal{A}}$ be the event such that a random bit b in $Game_x^{\mathcal{A}}$ is guessed correctly by $\mathcal{A}$. Thus, the advantage of $\mathcal{A}$ is the success probability to win $Game_x^{\mathcal{A}}$ given as $Adv_{\mathcal{A},Game_x}^{BACPI4} = Pr[Success_{Game_x}^{\mathcal{A}}]$. The following describes the games that the adversary $\mathcal{A}$ plays against the proposed scheme *BACPI*4:

$Game_0^{\mathcal{A}}$: This game is played by attacking *BACPI*4 that involves taking a random bit b at the start of the game $Game_0^{\mathcal{A}}$. From the "semantic security as defined in Definition 1," we obtain

$$Adv_{\mathcal{A}}^{BACPI4}(t_p) = |2Adv_{\mathcal{A},Game_0}^{BACPI4} - 1|. \tag{1}$$

$Game_1^{\mathcal{A}}$: This game is played by $\mathcal{A}$ by first calling the the *Execute* query to eavesdrop on the messages in the public channel, that is, $\text{Msg}_{\text{DAC}_1}$ and $\text{Msg}_{\text{DAC}_2}$ during DAC phase, and $\text{Msg}_{\text{GAC}_1}$ and $\text{Msg}_{\text{GAC}_2}$ during GAC phase. Calling of the *Test* query follows, whose outcome helps decide whether the *Reveal* query returns the original session keys or some random keys. The session key between device equipment nodes DN_1 and DN_2 is $\text{SK}_{ji} = H(g_j \cdot G_i||\text{RID}'_{N_i}||\text{RID}'_{N_j}||T_i||T_j) = H(g_i \cdot G_j||\text{RID}'_{N_i}||\text{RID}'_{N_j}||T_i||T_j) = \text{SK}_{ij}$ in the DAC phase and $\text{SK}_{ki} = H(v_k \cdot U_i||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2) = H(u_i \cdot V_k||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2) = \text{SK}_{ik}$ in GAC phase. It is noted that the session keys depend on both the temporary secrets r_i, r_j, s_i, w_k and the permanent secrets d_i, d_j, g_k, RID_{N_i}, RID_{N_j}, and RID_{G_k}. Additional security is provided by the "collision-resistant one-way hash function $H(\cdot)$" which protects all the temporary and permanent secrets. Thus the probability that $\mathcal{A}$ succeeds cannot rise only by eavesdropping on the message exchange as they cannot reveal the session keys $\text{SK}_{ij}(= \text{SK}_{ji})$ and $\text{SK}_{ik}(= \text{SK}_{ki})$. Therefore, $Game_0^{\mathcal{A}}$ and $Game_1^{\mathcal{A}}$ become semantically indistinguishable for the eavesdropping attack. Thus, it follows that

$$Adv_{\mathcal{A},Game_1}^{BACPI4} = Adv_{\mathcal{A},Game_0}^{BACPI4}. \tag{2}$$

$Game_2^{\mathcal{A}}$: This game is played by simulating *Hash* queries and execution of computational ECDDHP problem by $\mathcal{A}$ as an active attack. In DAC phase, the session key is derived as $\text{SK}_{ji} = H(g_j \cdot G_i||\text{RID}'_{N_i}||\text{RID}'_{N_j}||T_i||T_j) = H(g_i \cdot G_j||\text{RID}'_{N_i}||\text{RID}'_{N_j}||T_i||T_j) = \text{SK}_{ij}$. $\mathcal{A}$ can obtain RD_i, G_i and RD_j, G_j from $\text{Msg}_{\text{DAC}_1}$ and $\text{Msg}_{\text{DAC}_2}$. The known RD_i and RD_j can help derive derive the session key if the computational ECDDHP can be solved which results in obtaining the private rd_i and rd_j which in themselves are based on the secrets (r_i, r_j, d_i, d_j) unknown to $\mathcal{A}$. In addition, simulation of *Hash* queries is needed to compute g_i and g_j. The "collision-resistant one-way hash function $H(\cdot)$" protects over the private (r_i, r_j, d_i, d_j). In GAC phase, the session key is derived as $\text{SK}_{ki} = H(v_k \cdot U_i||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2) = H(u_i \cdot V_k||\text{RID}'_{N_i}||\text{RID}'_{G_k}||\text{TSP}_1||\text{TSP}_2) = \text{SK}_{ik}\mathcal{A}$ can obtain SD_i, U_i and WG_k, V_k from $\text{Msg}_{\text{GAC}_1}$ and $\text{Msg}_{\text{GAC}_2}$. The known SD_i and WG_k can help derive derive the session key if the computational ECDDHP can be solved which results in obtaining the private sd_i and wg_k which in themselves are based on the secrets (s_i, w_k, d_i, g_k) unknown to $\mathcal{A}$. In addition, simulation of *Hash* queries is needed to compute u_i and v_k. The "collision-resistant one-way hash function $H(\cdot)$" protects over the private (s_i, w_k, d_i, g_k). By excluding the *Hash* queries simulation and the computational ECDDHP from $Game_2^{\mathcal{A}}$, $Game_1^{\mathcal{A}}$ and $Game_2^{\mathcal{A}}$ can be made indistinguishable. Applying the birthday paradox to discover collisions in hashes along with advantage of solving ECDDHP can be shown with the following relationship:

$$|Adv_{\mathcal{A},Game_1}^{BACPI4} - Adv_{\mathcal{A},Game_2}^{BACPI4}| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(time_{poly}). \tag{3}$$

Except for guessing a bit to win the game $Game_2^{\mathcal{A}}$, all the queries discussed are executed by $\mathcal{A}$. Thus, $Adv_{\mathcal{A},Game_2}^{BACPI4} = \frac{1}{2}$. Applying triangular inequality over Equations (1)–(3) derives:

$$\frac{1}{2}.Adv_{\mathcal{A}}^{BACPI4}(t_p) = |Adv_{\mathcal{A},Game_0}^{BACPI4} - \frac{1}{2}| = |Adv_{\mathcal{A},Game_1}^{BACPI4} - Adv_{\mathcal{A},Game_2}^{BACPI4}| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p).$$

A minor multiplication by "a factor of 2" on both side gives the final result as: $Adv_{\mathcal{A}}^{BACPI4}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p)$. ∎

4.2 | Informal security analysis

4.2.1 | Replay attack

Let DN_i send a message $Msg^x_{DAC_1} : \langle RID'_{N_i}, RD_i, G_i, T_i, Sig_i\rangle$ in a session x which is captured by the adversary A and replayed in session y. DN_i generates r_i^x, T_i^x in session x and r_i^y, T_i^y in session y. The replay attack can only be successful if the hashes $H(rd_i^y||RID_{N_i}||pr_{N_i}||T_i^y) \stackrel{?}{=} RID^x_{N_i}$, $H(r_i^y||d_i||pr_{N_i}||T_i^y) \stackrel{?}{=} g_i^x$. For each hash computed, such a collision of hash values can be achieved with probability $1/2^{q/2}$ according to birthday attack, given that q is the number of bits in hash. For a sufficiently large value of q (≥ 160), the probability of such a collision is negligible since $r_i^x \neq r_i^y$ and $T_i^x \neq T_i^y$. Also, the verification of the signature Sig_i will fail due to mismatch of T_i^x and T_i^y. Thus, the message Msg_{DAC_1} cannot be replayed. By the same logic, replay attack cannot be successful on the messages Msg_{DAC_2}, Msg_{GAC_1}, Msg_{GAC_2} in the DAC and GAC phases and the proposed scheme is strongly resilient against this attack.

4.2.2 | Man-in-the-Middle (MiTM) attack

Any change to the parameters $RID'_{N_i}, RD_i, G_i, T_i$ in Msg_{DAC_1}, $RID'_{N_j}, RD_j, G_j, T_j$ in Msg_{DAC_2}, $RID'_{N_i}, SD_i, U_i, TSP_1$ in Msg_{GAC_1} and $RID'_{G_k}, WG_k, V_k, TSP_2$ in Msg_{GAC_2} by the adversary A can be easily identified through the verification of the signatures Sig_i, Sig_j, Sig_{D_i}, and Sig_{G_k} respectively for each of the messages in the DAC and GAC phases. Thus, the proposed scheme is resilient against man-in-the-middle attack.

4.2.3 | Impersonation attacks

Consider that the adversary A has captured the messages $Msg_{DAC_1} : \langle RID'_{N_i}, RD_i, G_i, T_i, Sig_i\rangle$ and $Msg_{DAC_2} : \langle RID'_{N_j}, RD_j, G_j, T_j, Sig_j\rangle$ in DAC phase and $Msg_{GAC_1} : \langle RID'_{N_i}, SD_i, U_i, TSP_1, Sig_{D_i}\rangle$ and $Msg_{GAC_2} : \langle RID'_{G_k}, WG_k, V_k, TSP_2, Sig_{G_k}\rangle$ in the GAC phase. A may try to impersonate the device equipment node DN and the gateway node GN.

1. *Device impersonation attack*: To impersonate the device, A fabricate the messages $Msg^A_{DAC_1} : \langle RID^{A'}_{N_i}, RD^A_i, G^A_i, T^A_i, Sig^A_i\rangle$ and $Msg^A_{DAC_2} : \langle RID^{A'}_{N_j}, RD^A_j, G^A_j, T^A_j, Sig^A_j\rangle$ in DAC phase and $Msg^A_{GAC_1} : \langle RID^{A'}_{N_i}, SD^A_i, U^A_i, TSP^A_1, Sig^A_{D_i}\rangle$ in GAC phase. This requires A to know the secrets r_i, r_j, s_i, d_i, d_j, ID_{N_i}, ID_{N_j}, n_i, and n_j which are not shared in any of the messages. Thus, the scheme is resilient against device impersonation attack.
2. *Gateway impersonation attack*: To impersonate the device, A fabricate the message $Msg^A_{GAC_2} : \langle RID^{A'}_{G_k}, WG^A_k, V^A_k, TSP^A_2, Sig^A_{G_k}\rangle$ which requires A to now the secrets w_k, g_k, ID_{G_k}, and m_k which are not shared in any of the messages exchange in public channels. Hence, the proposed scheme is resilient against gateway impersonation attack.

4.2.4 | Privileged-insider attack

The secret credentials RID_{N_i}, d_i are pre-loaded in DN_i during registration in Section 3.2.1. The secret credentials ID_{G_k}, RID_{G_k}, g_k for GN_k, ID_{F_l}, RID_{F_l} for FS_l and ID_{CS_y}, RID_{CS_y} for CS_y are exchanged via secure channels during registration processes described in Sections 3.2.2–3.2.4. Since none of these credentials are available in the public channels, privileged insider attack cannot be launched on the proposed scheme.

4.2.5 | Physical device node capture attack

The adversary A may physically capture a device equipment node and gain access to the contents $\{RID_{N_i}, d_i, H(\cdot), E_q(a,b), P, (pr_{N_i}, Pub_{N_i})\}$ its memory storage. However, the parameters $RID_{N_i}, d_i, (pr_{N_i}, Pub_{N_i})$ are specific to each device and not shared between devices. Thus, compromise of one device does not affect the other devices in the network.

4.2.6 | Ephemeral secret leakage (ESL) attack

The proposed scheme generates the session keys $SK_{ji} = H(g_j \cdot G_i||RID'_{N_i}||RID'_{N_j}||T_i||T_j) = SK_{ij}$ in the DAC phase and $SK_{ki} = H(v_k \cdot U_i||RID'_{N_i}||RID'_{G_k}||TSP_1||TSP_2) = SK_{ik}$ in GAC phase, which are dependent on the temporary secrets r_i, r_j, s_i, w_k and the permanent secrets d_i, d_j, g_k, RID_{N_i}, RID_{N_j}, and RID_{G_k}. Compromise of short term secrets does not affect the security of the session keys as the adversary A does not have access to the long term secrets. Similarly, compromise of long term secrets does not affect the security of the session keys as the adversary A does not have access to the short term secrets. The scheme is resilient against simultaneous compromise of long term and short term secrets as it is built considering the CK adversary model.

4.2.7 | Denial of Service (DoS) attack

The proposed scheme uses only hashing and ECC which are lightweight computations consuming very less resources. In addition, the timestamps T_i, T_j, TSP_1, and TSP_2 are verified for every message exchange to disallow multiple repeated messages from the same sender. This ensures resilience against DoS attacks. In addition, the DoS attack depletes server resources such as bandwidth and memory storage, resulting in obstruction of legitimate access to the servers. The fog server receives message from the gateway. If the transaction verification using Pub_{G_k} fails at the fog server, the entire message is dropped, and no further steps are taken. This safeguards the fog server's resources from being depleted by DoS attackers for partial block creation.

4.2.8 | Anonymity and untraceability

The messages exchanges use only the temporary identities RID'_{N_i}, RID'_{N_j}, and RID'_{G_k} which hide the real identities ID_{N_i}, ID_{N_j}, ID_{G_k} and the pseudo-identities RID_{N_i}, RID_{N_j}, and RID_{G_k}. Thus, none of the messages can be traced back to the real sending entity and anonymity of all entities is preserved.

4.2.9 | Forward secrecy

As described in Section 4.2.6, without simultaneously compromising the long term and short term secrets, it is computationally infeasible for an adversary to compute the session key in a particular session. Again, over the successive sessions, all the previous established session keys are unique and distinct due to different short term random secrets used in the session keys. Hence, the proposed scheme achieves the forward secrecy.[10,36]

> *Remark.* Privacy preservation is to ensure that if any intermediate parties involved in communication cannot know what data is being communicated. In our scheme, no communication uses intermediate parties. This is because when a device needs to communicate with another device or gateway, it establishes a direct session with that device/gateway and encrypts the data with the established session key. Hence, the privacy of the data is preserved. The private credentials used in the registration and authentication processes are the critical data in this proposed system. The data is never shared in any public channel directly.

5 | COMPARATIVE STUDY

In this section, we compare the performance of the proposed scheme with the recent existing competing schemes, such as the schemes of Baruah et al.,[9] Wang et al.,[10] Esfahani et al.,[14] and Garg et al.[15]

5.1 | Communication costs comparison

The communication cost for the schemes[9,10,14,15] has been computed for comparison with our scheme. The following are the considerations for the cost analysis: identities and random secrets are 160 bits, timestamps are 32 bits, a point

TABLE 3 Comparison of communication costs.

Protocol	No. of messages	Total cost (in bits)
9	3	1728
10	8	4432
14	5	2720
15	6	4352
Our scheme (DAC phase)	2	2176
Our scheme (GAC phase)	2	2176

TABLE 4 Average execution time (in milliseconds).

Primitive	Average time on Raspberry PI 3 (in milliseconds)	Average time on server (in milliseconds)
T_h	0.309	0.055
T_{exp}	0.228	0.072
T_{ecm}	2.288	0.674
T_{eca}	0.016	0.002
T_{enc}	0.018	0.001
T_{dec}	0.014	0.001
T_{bp}	32.084	4.603

Q = (x_q, y_q) o an elliptic curve $E_q(a, b)$ is taken as (160 + 160) = 320 bits, where each coordinate of the point is considered as 160 bits. It is assumed that a 160 bit ECC provides the same security level as a 1024 bit RSA system. The schemes of Baruah et al.,[9] Wang et al.,[10] Esfahani et al.,[14] and Garg et al.[15] consume communications costs of 1728, 4432, 2720, and 4352 bits respectively. The proposed scheme sends two messages costing 2176 bits in each of the DAC and GAC phases. A comparative study among the proposed scheme and other schemes on communication costs provided in Table 3 reveals that less or comparable communication costs as compared to the other schemes.

5.2 | Computation costs comparison

The computation cost for the schemes[9,10,14,15] has been computed for comparison with our scheme. The following are the considerations for the cost analysis: T_h, T_{ecm}, T_{exp}, T_{eca}, T_{enc}, T_{dec}, and T_{bp} are used as notations to denote the one way hash operation using SHA-256,[35] elliptic curve multiplication, modular exponentiation, elliptic curve addition, symmetric encryption, symmetric decryption operations using the AES-128[37] symmetric algorithm and the bilinear pairing operation. These operations have been executed on the widely known "multiprecision integer and rational arithmetic cryptographic library (MIRACL)"[38] as testbed experiments to obtain the average execution time of each of these operations as shown in Table 4. The proposed scheme uses hashing operation along with the elliptic curve addition and multiplication operations costing $5T_h + 5\ T_{ecm} + T_{eca}$ at the device side in both DAC and GAC phase and the same at the gateway side in the GAC phase.

Two scenarios have been considered for the the testbed experimentation using MIRACL library:

1. *Scenario 1*: The platform for a server is considered to have the following setting: "Ubuntu 18.04.4 LTS, with memory: 7.7 GiB, processor: Intel® Core$^{\hat{U}}$ i7-8565U CPU @ 1.80 GHz × 8, OS type: 64-bit and disk: 966.1 GB." The results of the "maximum, minimum and average run-time (in milliseconds) for each cryptographic primitive for 100 runs" are tabulated in Table 4.
2. *Scenario 2*: The platform for a user mobile device or a smart device is considered tp have the following setting: "Raspberry PI 3 B+ Rev 1.3, with CPU: 64-bit, Processor: 1.4 GHz Quad-core, 4 cores, Memory (RAM): 1 GB, and OS:

TABLE 5 Comparison of computation costs.

Protocol	Smart device	Server end
9	$5T_h + 5\ T_{XOR} + T_{exp}$	$6\ T_h + 6\ T_{XOR} + 2\ T_{exp}$
	≅ 1.773 ms	≅ 0.474 ms
10	$22T_h + 2\ T_{ecm} + 5\ T_{exp}$	$16\ T_h + T_{ecm} + 3\ T_{exp}$
	≅ 12.514 ms	≅ 1.77 ms
14	$7\ T_h + 4\ T_{XOR}$	$7\ T_h + 6\ T_{XOR}$
	≅ 2.142 ms	≅ 0.385 ms
15	$6\ T_{ecm} + 8\ T_h + 2\ T_{XOR}$	$6\ T_{ecm} + 8\ T_h + 4\ T_{XOR}$
	≅ 16.2 ms	≅ 4.484 ms
Our scheme	$5T_h + 5\ T_{ecm} + T_{eca}$	NIL
(DAC phase)	≅ 13.01 ms	
Our scheme	$5T_h + 5\ T_{ecm} + T_{eca}$	$5T_h + 5\ T_{ecm} + T_{eca}$
(GAC phase)	≅ 13.01 ms	≅ 3.647 ms

Ubuntu 20.04 LTS, 64-bit".[39] The results of the "maximum, minimum and average run-time (in milliseconds) for each cryptographic primitive for 100 runs" are also tabulated in Table 4.

A comparative study among the proposed scheme and other schemes on computational costs provided in Table 5 reveals that proposed scheme has less or comparable communication costs as compared to the other schemes.

Our work focuses on entity authentication and agreement of session key, which will be used for message encryption later. The cost of encryption would depend on the specific algorithm used for encryption, which is out of scope for the current work.

5.3 | Security and functionality features comparison

Table 6 provides a thorough comparison of the required security features for a scheme developed against the Industrial IoT network. It shows that Esfahani et al.'s scheme[14] do not support most of the security functionality features even though they have very low computation and communication costs. In addition, Baruah et al.'s scheme[9] does not resist ESL attack. The proposed scheme supports the features of anonymity, untraceability, forward secrecy and resisting impersonation attack, ESL attack, privileged insider attack, offline guessing attack.

The compared schemes are based on centralized servers that makes them vulnerable to single point of failure, resulting in loss of stored data from IoT sensor devices. On the other hand, our proposed scheme is decentralized making it fail-safe against single point of failure. Moreover, storing the data on the blockchain makes our system robust due to transaction and block verification process inherent in the blockchain technology, which amplifies trust on the data even if the blockchain nodes are trustless. Since the other schemes do not support blockchain, an fair comparison between the schemes is based on the operations performed during authentication.

It is worth noticing that the proposed scheme has quite low communication cost of only 2176 bits in each phase and comparable computation cost of $5T_h + 5T_{ecm} + T_{eca}$ at each entity in each phase, while also supporting the features of anonymity, untraceability, dynamic node addition phase and forward secrecy, and resisting impersonation, ESL, privileged insider and off-line guessing attacks, which are not supported in the other compared schemes.

6 | BLOCKCHAIN SIMULATION

The consensus mechanism for the proposed scheme using "Practical Byzantine Fault Tolerance (PBFT)" has been implemented on a platform with the environment details as follows: "Ubuntu 20.04.3 LTS (Focal Fossa), 64-bit OS with Intel(R)

TABLE 6 Comparison of security and functionality features.

Features	Baruah et al.[9]	Wang[10]	Esfahani et al.[14]	Garg et al.[15]	Proposed
$\mathcal{F}_1$	✓	✓	×	×	✓
$\mathcal{F}_2$	✓	✓	×	×	✓
$\mathcal{F}_3$	×	✓	×	N/A	✓
$\mathcal{F}_4$	×	✓	×	×	✓
$\mathcal{F}_5$	✓	×	×	✓	✓
$\mathcal{F}_6$	×	×	×	N/A	✓
$\mathcal{F}_7$	×	×	×	×	✓
$\mathcal{F}_8$	×	✓	×	×	✓
$\mathcal{F}_9$	×	×	✓	✓	✓
$\mathcal{F}_{10}$	✓	✓	✓	✓	✓
$\mathcal{F}_{11}$	✓	✓	✓	✓	✓
$\mathcal{F}_{12}$	✓	✓	✓	✓	✓
$\mathcal{F}_{13}$	✓	×	×	✓	✓
$\mathcal{F}_{14}$	✓	×	✓	×	✓
$\mathcal{F}_{15}$	×	✓	×	✓	✓
$\mathcal{F}_{16}$	×	×	×	×	✓

Note: $\mathcal{F}_1$: "anonymity"; $\mathcal{F}_2$: "untraceability"; $\mathcal{F}_3$: "user device revocation"; $\mathcal{F}_4$: "dynamic node addition"; $\mathcal{F}_5$: "user/node impersonation attack"; $\mathcal{F}_6$: "stolen smart card/mobile device attack"; $\mathcal{F}_7$: "ephemeral secret leakage (ESL) attack"; $\mathcal{F}_8$: "privileged insider attack"; $\mathcal{F}_9$: "replay attack"; $\mathcal{F}_{10}$: "man-in-the-middle attack"; $\mathcal{F}_{11}$: "mutual authentication"; $\mathcal{F}_{12}$: "unauthorized login detection"; $\mathcal{F}_{13}$: "Denial-of-Service (DoS) attack"; $\mathcal{F}_{14}$: "offline guessing attacks"; $\mathcal{F}_{15}$: "forward secrecy"; $\mathcal{F}_{15}$: "resistance to data loss due to single point of failure;" N/A: "not applicable in a scheme"; ✓: "a feature is supported in a scheme or resistant against the specified attack"; ×: "a feature is not supported in a scheme or it is not resilient against the specified attack."

Core(TM) i7-6820HQ CPU @2.70 GHz, 32GiB RAM." The gateway node creates transactions using the data received from the sensors in the smart factory. These transactions are sent to the fog servers in encrypted format. The fog servers create partial blocks of the received transactions, and sends them to the cloud server. The cloud server generates the full block by adding hash of previous block and computing new block hash. One of the cloud servers is elected as the leader and execution of consensus algorithm commences in the BCC.

The size of the full block as shown in Figure 5 is computed using the sizes of $\{BlockVer, Prev_BHash, MTR_{ParBlock_l}, Timestamp, Block_{Owner}, \{\text{Enc}_{\text{Pub}_{F_l}}(TX_k), \text{Enc}_{\text{Pub}_{F_l}}(\text{Sign}_{TX_k})\}, \text{Sign}_{ParBlock}, \text{Curr}_BHash\}$ as {32, 256, 256, 32, 160, num_{tr}*768, 160, 256} giving a total of 1152 + num_{tr}*768 bits. The simulation conducted in this work has considered synthetic data exchanged among the smart IoT device nodes in smart factories and between the smart IoT device nodes and the gateway nodes. The blocks are created out of the transactions collected by the fog servers. Thus, the real-time data has not been considered in this work. The proposed scheme does not require any new blockchain frameworks to be specifically designed. It can be used with any of the existing frameworks, such as Hyperledger Fabric, that support the PBFT. However, Ethereum 2.0 can also be used as the blockchain framework if the consensus algorithm is replaced with Proof-of-Stake (PoS) instead of PBFT.

For the simulation, a programming platform VS CODE 2019 has been used with Node.js language. Three scenarios taken into account for the simulation are as follows.

1. **Case 1:** Initially, the number of P2P nodes were considered to be 10 and the number of transactions per block were fixed to be 15. Then, the time for computation was noted as the number of blocks mined were increased from 10 to 50 and depicted in Figure 7.
2. **Case 2:** In the next scenario, the number of blocks mined was fixed at 10 and each block consisted of 25 transactions. The number of P2P nodes used to mine these fixed no of blocks were varied from 5 to 25 and the computation time is depicted in Figure 8.

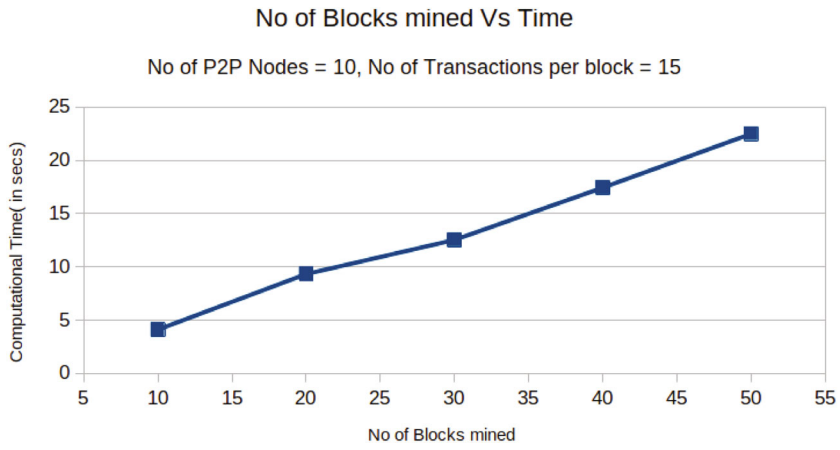

FIGURE 7 Blockchain simulation results: No. of blocks mined versus time.

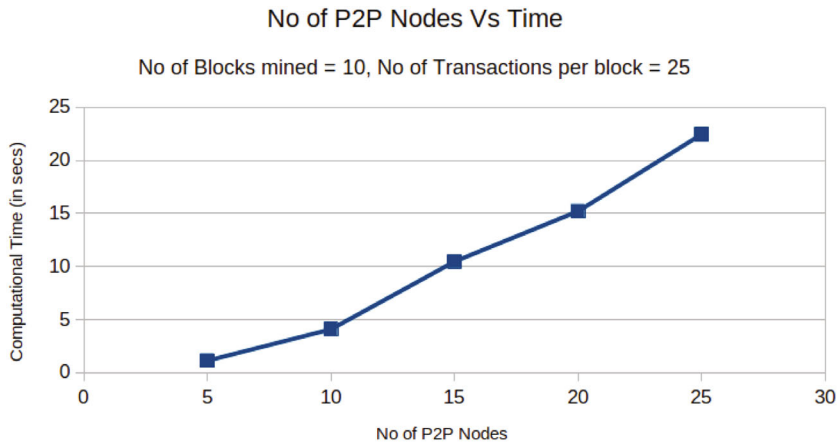

FIGURE 8 Blockchain simulation results: No. of P2P nodes versus time.

3. **Case 3:** In the last scenario, 10 P2P nodes creating 10 blocks to be mined are considered. The number of transactions in each block is varied from 10 to 50 and depicted in Figure 9.

The results of the blockchain simulation show that the computation time for mining increases linearly with a linear increase in the number of P2P nodes in the blockchain network, a linear increase in the number of blocks, and a linear increase in the number of transactions in each block. It is also noted that the transaction throughput (transactions processed per second) of a blockchain-based system can be calculated as follows:

$$\text{TPS} = \frac{n_{BTx} \times n_{BM}}{T_{\text{comp}}},$$

where n_{BTx} is the number of transactions per block, n_{BM} is the number of blocks mined, and T_{comp} is the computation time in seconds. Due to this, the transaction throughput measured as the number of transactions per second also increases linearly when transactions per block, number of blocks and computational time increase linearly.

7 | OPEN RESEARCH DIRECTIONS

The traditional approach to adding security to any IoT environment is based on developing and executing rules, which when violated showcase a loss of security features. Usage of artificial intelligence/machine learning (AI/ML) techniques

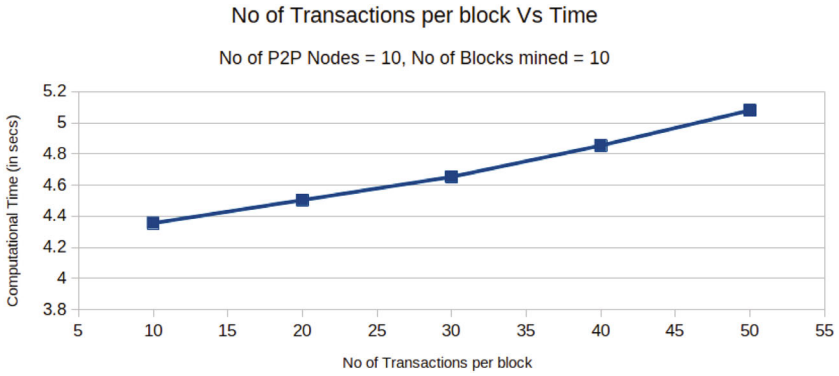

FIGURE 9 Blockchain simulation results: No. of transactions per block versus time.

improves the identification of vulnerabilities in the network, identifying threats and attacks, and analytics on the threat data which help in improving the model by using a feedback mechanism. Xiao et al.[40] identified that authentication, access control, malware detection and task offloading techniques are four major major areas of IoT security, where learning techniques such as "support vector machines (SVM)," "Naive Bayes algorithm," and "neural networks" can be used for access control. Many techniques devised under supervised, unsupervised, deep, reinforcement and ensemble learning can be used to augment device authentication, malware detection and alleviate DoS attacks by a systematic approach involving data collection, pre-processing, model selection, data conversion, training model, testing model, and deployment of model after evaluation. Multiple models may also be combined and enhanced with decentralized frameworks.[41] The data collected from the IoT devices may be structured, semi-structured or unstructured. Usage of AI with IoT can improve accuracy rate and operational efficiency through predictive analysis. Involving blockchain technology over AI/ML techniques increases the trust in the system as the analytical techniques are applied over data in the blockchain which has only been added after thorough transaction verification and consensus.[42]

8 | CONCLUSION

This research work proposes a novel access control scheme using private blockchain that allows sensitive sensor data from the equipment in the industrial factories to be verified and validated thoroughly before storing on the blockchain. This storage on blockchain only occurs after the device nodes and the gateway nodes successfully establish a session to allow access of data from one another using the DAC and GAC phases of the scheme. Hence, the data can now be trusted to be used easily for further processing and analysis through various techniques in Big data analytics. A novel architecture for Industry 4.0 has been proposed. The scheme has been analyzed through rigorous security analysis using the formal ROR model and informal security analysis. A comparative study that reveals its resilience against several security attacks such as privileged insider attack, ESL attack, replay attack and man-in-the-middle attack through the presence of strong security features such as forward secrecy, dynamic node addition, dynamic node revocation, anonymity, and untraceability. The scheme achieves such high security objectives with minimal communicational and comparable computational costs as has been studied using the testbed experimentation with MIRACL library.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and the editor for their valuable suggestions and comments, which helped to improve this article's presentation and technical quality. This work was supported by "Design and Development of a Unified Blockchain Framework for offering National Blockchain Service" project, under the Ministry of Electronics & Information Technology, New Delhi, Government of India (No. 4(4)/2021-ITEA) and by "Design of Blockchain- Based Authentication and Access Control Protocols in Internet of Vehicles and Internet of Things Deployment" project under the Ripple Centre of Excellence (CoE) Scheme, CoE in Blockchain, IIIT Hyderabad, India (No. IIIT/R&D Office/Internal Projects/001/2019). This research was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2020R1I1A3058605.

CONFLICT OF INTEREST STATEMENT

The authors declare that they do not have any commercial or associative interest that represents the conflict of interest in connection with the work submitted.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Ashok Kumar Das https://orcid.org/0000-0002-5196-9589
Neeraj Kumar https://orcid.org/0000-0002-3020-3947

REFERENCES

1. Ghobakhloo M. Industry 4.0, digitization, and opportunities for sustainability. *J Clean Prod*. 2020;252:119869.
2. Nigam V, Talcott C. Formal security verification of Industry 4.0 applications. *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE; 2019:1043-1050.
3. Zhang Q, Liao B, Yang S. Application of blockchain in the field of intelligent manufacturing: theoretical basis, realistic plights, and development suggestions. *Front Eng Manag*. 2020;7(4):578-591.
4. Fernandez-Carames TM, Fraga-Lamas P. A review on the application of blockchain to the next generation of cybersecure Industry 4.0 smart factories. *IEEE Access*. 2019;7:45201-45218.
5. Dolev D, Yao A. On the security of public key protocols. *IEEE Trans Inf Theory*. 1983;29(2):198-208.
6. Wang D, Li W, Wang P. Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks. *IEEE Trans Industr Inform*. 2018;14(9):4081-4092.
7. Canetti R, Krawczyk H. Universally composable notions of key exchange and secure channels. *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*. Springer; 2002:337-351.
8. Messerges TS, Dabbish EA, Sloan RH. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans Comput*. 2002;51(5):541-552.
9. Baruah B, Dhal S. An efficient authentication scheme for secure communication between industrial IoT devices. *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE; 2020:1-7.
10. Wang C, Wang D, Xu G, He D. Efficient privacy-preserving user authentication scheme with forward secrecy for Industry 4.0. *Sci China Inf Sci*. 2022;65(1):1-15.
11. Zagrouba R, AlAbdullatif A, AlAjaji K, et al. Authenblue: a new authentication protocol for the Industrial Internet of Things. *Comput Mater Contin*. 2021;67(1):1103-1119.
12. Bodkhe U, Tanwar S, Parekh K, et al. Blockchain for Industry 4.0: a comprehensive review. *IEEE Access*. 2020;8:79764-79800.
13. Javaid M, Haleem A, Pratap Singh R, Khan S, Suman R. Blockchain technology applications for Industry 4.0: a literature-based review. *Blockchain Res Appl*. 2021;2:100027.
14. Esfahani A, Mantas G, Matischek R, et al. A lightweight authentication mechanism for M2M communications in industrial IoT environment. *IEEE Internet Things J*. 2019;6(1):288-296.
15. Garg S, Kaur K, Kaddoum G, Choo K-KR. Toward secure and provable authentication for internet of things: realizing Industry 4.0. *IEEE Internet Things J*. 2020;7(5):4598-4606.
16. Kolluru KK, Paniagua C, van Deventer J, Eliasson J, Delsing J, DeLong RJ. An AAA solution for securing industrial IoT devices using next generation access control. *IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE; 2018:737-742.
17. Shuai M, Xiong L, Wang C, Yu N. A secure authentication scheme with forward secrecy for Industrial Internet of Things using Rabin cryptosystem. *Comput Commun*. 2020;160:215-227.
18. Gupta DS, Islam SH, Obaidat MS, Vijayakumar P, Kumar N, Park Y. A provably secure and lightweight identity-based two-party authenticated key agreement protocol for IIoT environments. *IEEE Syst J*. 2021;15(2):1732-1741.
19. Li Y, Cheng Q, Shi W. Security analysis of a lightweight identity-based two-party authenticated key agreement protocol for IIoT environments. *Secur Commun Netw*. 2021;2021:5573886.
20. Rangwani D, Sadhukhan D, Ray S, Khan MK, Dasgupta M. A robust provable-secure privacy-preserving authentication protocol for Industrial Internet of Things. *Peer Peer Netw Appl*. 2021;14(3):1548-1571.
21. Tange K, De Donno M, Fafoutis X, Dragoni N. A systematic survey of Industrial Internet of Things security: requirements and fog computing opportunities. *IEEE Commun Surv Tutor*. 2020;22(4):2489-2520.
22. Yang Z, He J, Tian Y, Zhou J. Faster authenticated key agreement with perfect forward secrecy for Industrial Internet-of-Things. *IEEE Trans Industr Inform*. 2020;16(10):6584-6596.
23. Srinivas J, Das AK, Wazid M, Kumar N. Anonymous lightweight chaotic map-based authenticated key agreement protocol for Industrial Internet of Things. *IEEE Trans Dependable Secure Comput*. 2020;17(6):1133-1146.
24. Vinoth R, Deborah LJ, Vijayakumar P, Kumar N. Secure multifactor authenticated key agreement scheme for Industrial IoT. *IEEE Internet Things J*. 2021;8(5):3801-3811.

25. Far HAN, Bayat M, Das AK, Fotouhi M, Pournaghi SM, Doostari M-A. LAPTAS: lightweight anonymous privacy-preserving three-factor authentication scheme for WSN-based IIoT. *Wirel Netw*. 2021;27(2):1389-1412.
26. Raque F, Obaidat M, Mahmood K, Ayub MF, Ferzund J, Chaudhry SA. An efficient and provably secure certificateless protocol for Industrial Internet of Things. *IEEE Trans Industr Inform*. 2022;18:8039-8046.
27. Karati A, Islam SH, Karuppiah M. Provably secure and lightweight certificateless signature scheme for IIoT environments. *IEEE Trans Industr Inform*. 2018;14(8):3701-3711.
28. Rezaeibagha F, Mu Y, Huang X, Yang W, Huang K. Fully secure lightweight Certificateless signature scheme for IIoT. *IEEE Access*. 2019;7:144433-144443.
29. Zhang Y, Deng RH, Zheng D, Li J, Wu P, Cao J. Efficient and robust certificateless signature for data crowdsensing in cloud-assisted Industrial IoT. *IEEE Trans Industr Inform*. 2019;15(9):5099-5108.
30. Avoine G, Canard S, Ferreira L. Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In: Jarecki S, ed. *Cryptographers' Track at the RSA Conference*. Springer; 2020:199-224.
31. Aghili SF, Jolfaei AA, Abidin A. SAKE+: strengthened symmetric-key authenticated key exchange with perfect forward secrecy for IoT; 2020. https://ia.cr/2020/778
32. Ferreira L. Cryptanalysis of a "Strengthened" key exchange protocol for IoT, or when SAKE^{+} turns out to be SAKE^{-}; 2020. https://ia.cr/2020/839
33. Fan Q, Chen J, Shojafar M, Kumari S, He D. SAKE*: a symmetric authenticated key exchange protocol with perfect forward secrecy for Industrial Internet of Things. *IEEE Trans Industr Inform*. 2022;18:6424-6434.
34. Koblitz N, Menezes A, Vanstone S. The state of elliptic curve cryptography. *Des Codes Cryptogr*. 2000;19(2-3):173-193.
35. May WE. Secure Hash Standard. 2015, FIPS PUB 180-1. National Institute of Standards and Technology (NIST), U.S. Department of Commerce; August 2015. Accessed January 2022. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
36. Ma C-G, Wang D, Zhao S-D. Security flaws in two improved remote user authentication schemes using smart cards. *Int J Commun Syst*. 2014;27(10):2215-2227.
37. Advanced Encryption Standard. 2001, FIPS PUB 197. National Institute of Standards and Technology (NIST), U.S. Department of Commerce; November 2001. Accessed June 2020. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
38. MIRACL cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library; 2020. Accessed July 2020. https://github.com/miracl/MIRACL
39. Raspberry Pi 3 Model B+; 2020. Accessed April 2020. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
40. Xiao L, Wan X, Lu X, Zhang Y, Wu D. IoT security techniques based on machine learning: how do IoT devices use AI to enhance security? *IEEE Signal Process Mag*. 2018;35(5):41-49.
41. Wu H, Han H, Wang X, Sun S. Research on artificial intelligence enhancing Internet of Things Security: a survey. *IEEE Access*. 2020;8:153826-153848.
42. Mohanta BK, Jena D, Satapathy U, Patnaik S. Survey on IoT security: challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet Things*. 2020;11:100227.

How to cite this article: Vangala A, Das AK, Kumar N, Vijayakumar P, Karuppiah M, Park Y. Designing access control security protocol for Industry 4.0 using Blockchain-as-a-Service. *Security and Privacy*. 2024;7(2):e362. doi: 10.1002/spy2.362