Cluster Computing (2022) 25:451–468
https://doi.org/10.1007/s10586-021-03420-2

Blockchain-assisted authentication and key agreement scheme for fog-based smart grid

Ashish Tomar[1] · Sachin Tripathi[1]

Received: 23 May 2021 / Revised: 31 August 2021 / Accepted: 15 September 2021 / Published online: 25 September 2021

Abstract
In recent times, the research works on the integration of fog computing with blockchain to address the issues such as higher latency, single point of failure, and centralization have expanded considerably. However, only a few works have been done focusing on authentication and key establishment for blockchain-based smart grid (SG) under fog environment. Thus, this paper introduces a mutual authentication and key agreement scheme for blockchain and fog computing based SG environment. Unlike the existing schemes that depend on single trusted authorities for storage and computation tasks, the proposed scheme reduces this dependency by creating a blockchain-based distributed environment assisted by cloud servers and fog nodes. In addition, a secure and shared key is established among smart meter, fog node, and cloud server to achieve message confidentiality between them. A detailed formal and informal security analysis proves that the proposed protocol is secure under the RoR model and achieves the predefined security goals. The blockchain and cryptographic operations are evaluated using hyperledger fabric and cryptographic libraries, respectively. Finally, the performance analysis and comparative study show that the proposed scheme with some additional features is efficient in computational and communication costs.

Keywords Mutual authentication · Fog computing · Blockchain · Smart grid · Key agreement

1 Introduction

With the emergence of Internet of Things (IoT) technology, the interconnection between smart devices and sensors have become common nowadays. These devices have communication as well as computing capabilities that enable them to process and exchange data with other devices or entities. Therefore, IoT applications have been utilized rapidly in pervasive environment, where both opportunities and challenges prevail [1, 2]. Smart grid (SG) is one such IoT application to improve the sustainability, flexibility, security and reliability of traditional power grid [3]. Smart grid integrates some modern technologies of computing, communication, and sensing with power grid to provide active electrical network management. In addition, SG operations are more efficient, secure, stable, and reliable due to distributed intelligence and better automation and support various heterogeneous applications such as vehicle-to-grid service, voltage regulation, smart energy delivery, demand and response management, renewable energy resources integration, frequency support, etc. [4]. However, along with the SG popularity, the number of devices and customers in a SG environment have also shown explosive growth. Thus, SG environment faces several issues, such as higher computational and storage requirements, increasing latency, and maintaining quality of service (QoS) [5]. To solve these issues, cloud computing based solutions bring additional challenges like higher network bandwidth and latency due to the distance between smart meter and cloud server. Therefore, it makes it difficult to handle emergency situations in real-time [6]. Recently, few attempts are made to bring services of the cloud to the network edge, for instance, fog node deployment across the edge of the network.

Fog computing has developed as a possible paradigm to supplement cloud services in order to overcome the issue

✉ Ashish Tomar
ashishtomar244@gmail.com

[1] Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, India

of significant latency between IoT or mobile devices and the cloud. Fog extends the standard cloud to the network's edge, where delay-sensitive applications can take advantage of fog's closeness [7, 8]. In fog computing, the computational and storage tasks are moved towards the network edge devices, thereby reducing the computational overhead, bandwidth, and transmission time [6, 9]. In addition, fog offers many benefits such as quick response to delay-sensitive applications, data aggregation, data protection, context-aware and location-aware services. Table 1 shows the basic difference between cloud and fog. Furthermore, there have also been efforts to integrate blockchain in the fog environment to achieve blockchain features like distributed data storage, data immutability, data audit, and fault tolerance in fog computing infrastructure [5, 10–12].

However, security still remains the biggest issue to be addressed in the SG environment. As the saying goes, none of the system is entirely secure, and the same goes for the SG system. In smart grid network, Information and Communication Technologies (ICT) is used for two way communications to provide efficient and reliable data transfer operation. Despite many benefits of ICT, it makes SG environment unsafe against numerous security attacks like impersonation, man-in-the-middle attack, replay attack, spoofing, etc [4]. Moreover, an attacker can exploit the openness of ICT to capture the data flow, thereby creating an imbalance in demand and supply and revealing private data through long time analysis. In a fog based architecture, the attackers can exploit various characteristics of fog computing like mobility, heterogeneity, location awareness, and geo-distribution to perform malicious activities [6].

Mutual authentication with the key agreement is considered a robust measure to deal with the security vulnerabilities like security attacks and privacy issues in SG communication. PKI-based mutual authentication schemes are prevalent to maintain the trust between entities in a communication system. However, resource-constrained devices such as smart meters make them unsuitable for a SG system. In other words, the certificates issued by a single certificate authority (CA) are large in size that incur high communication and computation cost. Identity-based cryptography (IBS) based schemes could solve the problem of certificate management. However, it needs the leakage of one entity's actual identity to another participating entity. Since a fog node is a semi-trusted device and prone to attacks, it is not appropriate to store real identities in a fog node [13]. Ring signature-based schemes could be a possible solution to avoid identity sharing with the verifier [14, 15]. But it has some inevitable disadvantages such as untraceabilty of a malicious user, high computation and communication cost, no flexible participation, etc. Group signature schemes can be used to achieve traceability and flexible participation, but high computation and communication cost remains an issue.

Overall, SG environment has many advantages over the traditional power grid, such as (1) better flexibility, reliability, and sustainability, (2) provides active electrical network management, (3) grid operations are efficient and stable due to distributed intelligence and better automation support, (4) supports additional features like—smart energy delivery, voltage regulation, frequency support, etc. Despite the advantages, the SG environment also has some limitations. With the increase of the number of devices, the amount of generated data also increases that causes scalability-related issues such as higher storage and

Table 1 Difference between cloud and fog

Parameter	Cloud	Fog
Proximity	Global	Local
Latency	High	Low
Resources	Practically unlimited	Limited memory, compute, and storage
Service type	Software as a service, platform as a service, infrastructure as a service	Software defined networking, content deliver, data protection and security, device management
Deployment	Centralized	Decentralized
Communication overhead	Raw data (necessary and unnecessary communication through the core)	Refined and necessary communication through the core
Nodes	Servers	Routers, switches, gateways, servers, access points
Main content generator	Human	Devices and sensors
Data and communication security	Undefined, depends on service	Adds additional layer of security before sending data to the core
Operating expenses	High	Low

computational cost, higher latency, etc. The communication technologies used in SG environment are open in nature that makes SG vulnerable to various security attacks. Moreover, the SG environment follows a centralized architecture, where a centralized entity (trusted authority) is responsible for storage and computation tasks. It causes problems such as single point of failure, inefficient data handling, etc. Therefore, security, efficiency, and scalability are still big challenges to solve in the SG environment.

Thus, in the proposed scheme, we integrate blockchain technology in the SG with the assistance of fog and cloud servers to create a distributed environment. The computational and storage tasks are divided into multiple cloud servers and fog servers to avoid single point of failure, higher latency, etc. Cloud servers and fog servers act as blockchain peers to achieve coordination between them. In addition, we propose a three-party mutual authentication and key agreement scheme to achieve secure communication.

1.1 Motivation and contributions

As discussed, due to the resource constrained nature of IoT devices, the existing authentication and key agreement schemes for SG systems face many security and performance challenges. For example, the increasing number of devices in SG environment creates scalability related issues. To solve this, some researchers have utilized edge or fog devices to handle the computation and storage tasks [5, 6]. However, fog devices are not fully trusted; therefore, sharing secret information with them, such as identity, is not feasible. The over-dependence on a single trusted authority is another issue faced by the currently existing schemes. The centralized architecture causes many problems like single point of failure, centralized data storage that limits the SG environment's growth. To deal with the limitations of existing schemes, Wang et al. [5] incorporated blockchain in the SG edge computing environment to achieve low-cost revocation and key updation. However, their scheme suffers from blockchain scalability issues, i.e., with the increase in the number of devices, blockchain size also increases, making edge devices unable to perform blockchain operations as they are also resource-constrained and not capable enough to handle a global blockchain.

Therefore, to address the problems such as scalability, over-dependence on a single trusted authority, high storage and communication cost, and security vulnerabilities, we have contributed the following in the proposed scheme:

1. We integrate blockchain and fog computing into a SG environment and propose a three-layer architecture. In the proposed architecture, we follow the hyperledger fabric blockchain network model to achieve the coexistence of multiple cloud servers and fog nodes in a single blockchain network. It reduces the blockchain overhead from fog nodes and moves it towards the cloud server by letting cloud servers interact with the global blockchain.
2. To achieve better coordination and secure communication between smart meters, fog nodes, and cloud servers, we propose a three-party key agreement scheme that provides mutual authentication and common key establishment, called as BAIoG.
3. A detailed theoretical and informal security analysis confirms that the proposed BAIoG protocol fulfills all the predefined security goals.
4. The blockchain operations are evaluated using the hyperledger fabric platform. We deploy a basic business blockchain network to a single organization fabric instance and evaluate the running times of block-chain transactions using chaincode installed on the peers. Furthermore, we utilize cryptographic libraries to evaluate different cryptographic operations, which shows that our scheme performs better than the most of the existing schemes.

The remaining paper is organized as follows: Sect. 2 illustrates the existing research works related to authenticated key agreement schemes for SG environment, followed by Sect. 3, which describes background knowledge about the proposed work. Section 4 describes the steps of the proposed BAIoG protocol. Furthermore, in Sects. 5 and 6, we explain the security proof and analysis and performance analysis of the proposed work. Finally, Sect. 7 concludes the proposed scheme.

2 Related work

In the last few years, many authentication and key agreement schemes have been surfaced to deal with the privacy and security problems in the SG environment. In 2011, Wu et al. [16] introduced key agreement in the SG communication system by proposing a key management protocol with fault-tolerance, based on the integration of ECC public key technique and symmetric key technique. They claimed that their scheme is scalable and provides security against replay and man-in-the-middle attacks. Later in the same year, Fouda et al. [17] presented a Diffie–Hellman key exchange and hash function based message authentication protocol. They asserted that their scheme achieves key establishment and fulfills the security needs of a SG network. Xia and wang [18] analyzed the security of Wu et al.'s [16] scheme and proved that their scheme could not withstand man-in-the-middle attacks. They further

proposed third party based key distribution protocol in which third party works as a lightweight LDAP server. Due to the absence of PKI, their scheme attains low communication and computation cost. However, Park et al. [19] pointed out that Xia and Wang's protocol is unsafe against impersonation attack and unknown key-share attack.

Hereafter, in 2012, Liu et al. [20] introduced a key graph-based key management scheme for SG advanced metering infrastructure. They designed protocol to support multicast, broadcast, and unicast modes and claimed forward and backward security. Unfortunately, Wan et al. [21] illustrated that Liu et al.'s [20] protocol cannot withstand desynchronization attacks and faces scalability issues. Wan et al. [21] also presented a key management protocol with better scalability and security. However, their scheme is based on bilinear pairings, which results in high computational cost.

To achieve meter anonymity in a SG environment, Tsai and Lo [22], in 2015, combined identity-based encryption and identity-based signature and presented an anonymous key distribution scheme. But, their scheme is not safe against some attacks such as offline password guessing attacks, privileged insider attack and does not ensure meter privacy and session key security [23, 24]. Hereafter, Odelu et al. [24] proposed a new authentication and key agreement protocol and demonstrated that their scheme is secure under CK-adversary model. However, their scheme could not resist impersonation attack [25]. Furthermore, both Tsai and Lo [22] and Odelu et al. [24] schemes are bilinear pairings based therefore incur high computation cost.

To satisfy lightweight needs, He et al. [26] proposed a lightweight key distribution scheme using elliptic curve cryptography. Their scheme provides both mutual authentication and smart meter anonymity. In addition, Abbasinezhad-Mood and Nikooghadam [27] gave a self-certified ECC based key distribution scheme. They claimed that their scheme satisfies anonymity of smart meter and removes key escrow problem and certificate management problem. But their scheme does not support dynamic meter addition and is vulnerable to replay attack [23]. Zhang et al. [28] presented a lightweight key agreement scheme with privacy for meters. They verified the scheme under the RoR model and implemented it on the actual hardware prototype. But their scheme is found to be susceptible to impersonation attacks due to private data transmission over a public network [29]. Furthermore, Garg et al. [4] combined "Fully Hashed Menezes-Qu-Vanstone (FHMQV) key exchange mechanism" with ECC to design a mutual authentication and key agreement protocol. Jo et al. [23] used schnorr's signature to provide efficient authentication and key establishment. Their scheme supports dynamic meter addition as well as provides security against different attacks. However, Garg et al. [4] and Jo et al.'s [23] schemes need two rounds of communication, making the key agreement process complicated [30]. Deng and Gao [31] proposed a certificate less key agreement scheme to avoid certificate management and key escrow problems. They proved their scheme's security under the eCK model and claimed that their scheme performs better than existing schemes. Khan et al. [32] proposed an authenticated lightweight key agreement protocol for SG. They claimed that their scheme avoids key escrow issue and provides more privacy and security features. Badar et al. [33] proposed a PUF (physical uncloneable function) based authentication and key agreement for power supply line surveillance in SG. Sadhukan et al. [29] presented a mutual authentication and key negotiation scheme based on ECC with trifling operations between consumer and substation in a SG environment. They formally verified their scheme under the RoR model and claimed that their scheme is secure against all the possible security threats. Zhang et al. [34] designed an efficient and secure Chebyshev polynomial algorithm using square matrix-based binary exponentiation algorithm. Based on this, they proposed authentication and key agreement protocol with efficient energy utilization for SG environment.

Recently few attempts have been made to leverage the advantages of fog or edge computing and blockchain technology to achieve scalable and distributed environment for a SG system. Zhu et al. [37] gave an idea of integrating fog infrastructure into a SG environment. They also designed an anonymous authentication scheme based on blind signature for fog based smart grid. Khan et al. [38] presented a fog-based secure privacy-enabled data aggregation scheme with fault-tolerance. Their scheme achieves data privacy through Boneh–Goh–Nissam (BGN) cryptosystem and source authentication through Elliptic Curve Digital Signature Algorithm (ECDSA). Chen et al. [35] proposed a lightweight mutual authentication and key exchange scheme for edge-based SG environment. They used one way hash functions, XOR operations, and elliptic curve operations to achieve lightweight feature. Zhang et al. [39] introduced the integration of blockchain in the SG environment. They proposed a decentralized keyless signature scheme to achieve efficient authentication for a smart grid communication network. Afterward, Wang et al. [30] proposed a blockchain-based mutual authentication and key agreement protocol for the SG. However, in their scheme, it is not specified which type of blockchain is utilized because if consortium blockchain is utilized, then access rights must be defined; else, if a public blockchain is used, then anyone can access the private data stored in the blockchain. Chen et al. [40] integrated blockchain and fog computing in a SG environment and proposed an anonymous data aggregation scheme to achieve secure and

efficient data collection in smart grid. They claimed that their scheme achieves strong user privacy and confidentiality, integrity, and authenticity of data. Naseer et al. [41] used multiple service providers as blockchain nodes and proposed a decentralized environment to avoid single point of failure in the SG. They also proposed a trusted access control scheme for secure data transmission. However, in their scheme, computational and storage tasks are still centralized to the service provider.

Wang et al. [5] recently integrated blockchain with edge computing and proposed authentication and key management scheme with better scalability and low-cost revocation. They employed a consortium blockchain in edge computing infrastructure in which only edge devices join the network, i.e., edge devices work as blockchain peers. However, as the number of devices increases in the smart grid, the blockchain also increases in size, requiring more computational and storage power. Since edge devices are resource-constrained, it is impracticable to deploy a large blockchain on edge devices. In addition, their scheme is vulnerable to impersonation and key material change attack [42]. Therefore, to deal with the problems of, efficiency and scalability, this paper proposes a new authentication and key agreement scheme for a blockchain based SG fog infrastructure. Table 2 shows the feature comparison between the proposed scheme and some latest existing schemes.

3 Preliminaries

In this section, we discuss the background knowledge for the proposed scheme.

3.1 Blockchain technology

Blockchain is a new distributed ledger technology that uses cryptographic hashing and decentralization to achieve immutability and transparency among the connected blocks of transactions. The transactions are defined in the smart contract and performed through mutual consensus of participating peers. On the basis of access requirements, blockchain can be categorized into public, private, and consortium blockchain. In our work, We employ consortium open-source blockchain platform hyperledger fabric. It has the following components:

Table 2 Security feature comparison of fog and blockchain related works

Existing works	Three-layer architecture	Multi-channel support (blockchain)	Three-party key agreement	No sensitive data in fog node	Consortium blockchain	Distributed database storage	Resistance against security attacks
Wang et al.[5]	×	×	×	✓	✓	✓	×
Ma et al. [6]	×	–	✓	✓	–	×	✓
Wang et al. [30]	×	×	×	–	×	✓	✓
Chen et al.[35]	×	–	×	✓	–	×	✓
Jia et al. [36]	×	–	✓	✓	–	×	✓
Zhu et al. [37]	×	–	×	×	–	×	✓
Khan et al. [38]	×	–	×	✓	–	×	✓
Zhang et al. [39]	×	×	×	–	✓	✓	✓
Chen et al. [40]	✓	×	×	–	×	✓	✓
Naseer et al. [41]	×	×	×	–	×	✓	✓
Ours	✓	✓	✓	✓	✓	✓	✓

1. *Peers* Peers take part in the consensus process and holds the ledger and smart contract. Therefore, they are the basic building block for the formation of a fabric network.
2. *Ledger* It has two distinct, though related, parts—a blockchain and a world state, where blockchain consists of transaction log stored in an immutable chain of data blocks, and the world state holds the current values of records in the form of key-value pairs. The Ledger is distributed between participating peers.
3. *Orderer* In the fabric network, the orderer node carries out the tasks such as transaction ordering, creating blocks and distributes them to participating nodes.
4. *Smart Contract* The smart contract contains the logic for transactions in the form of executable code. Multiple smart contracts are grouped inside a chaincode, which is later deployed on peers.
5. *Channel* A channel is a medium of communication between participants of the consortium. A fabric network can have multiple channels, and a peer can be part of various channels.

3.2 Network model and assumptions

As shown in Fig. 1, the proposed three-layer smart grid fog architecture has the following components: Trusted Authority (TA), Cloud Server (CS), Fog Node (FN), Smart Meter (SM), Blockchain (BC).

1. *TA* Trusted authority is a government electricity board or a private service provider. It is a trusted entity responsible for registering SG and fog nodes and provides authentication parameters to registered entities. TA works as an administrator for fabric network and is responsible for chaincode installation on fabric peers. The data in the cloudchain can be added or updated by TA using API.
2. *CS* Cloud server is a trusted entity that acts as a peer of the blockchain. It is responsible for verifying smart meters and fog nodes through blockchain during authentication and key agreement step. It is the part of two fabric channels, one with other cloud servers and another one with the fog nodes. The blockchain formed by multiple cloud servers is addressed as 'cloudchain' throughout our scheme.
3. *FN* Fog node acts as a peer in the blockchain formed by multiple fog nodes and a cloud server (addressed as fogchain). It works as a utility controller or aggregator that supplies service delivery and timely data analysis. It also facilitates smart meters to add their data into the blockchain and later verifies them during the authentication and key agreement phase. It provides authentication messages between SM and CS, and a shared key is established between SM, FN, and CS.
4. *SM* Smart meter is a device inside a smart home responsible for sending energy utilization data to the nearest fog node. It first registers with TA and later adds verification parameters to fogchain through API for later authentication.

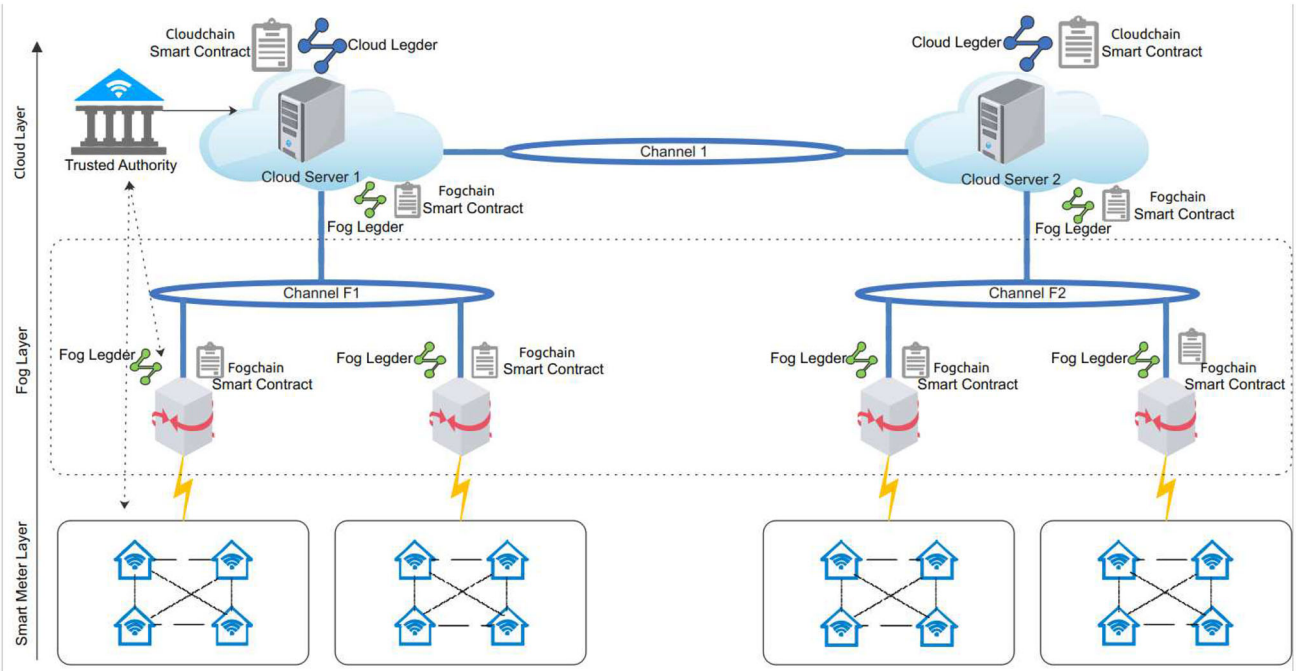

Fig. 1 Proposed system model

5. *BC* The blockchain operations are demonstrated by employing the consortium blockchain platform Hyperledger Fabric. It holds pseudo identities, identities, and other verification parameters related to smart meters and provides distributed access to all the participating cloud servers and fog nodes. It also provides services to handle different transactions performed by clients and peers.

3.3 Security goals

A secure and efficient mutual authentication and key agreement scheme for SG environment should achieve following security goals [5].

1. *Mutual Authentication* All the participants need to be validated by each other. Therefore, an authentication scheme should support mutual authentication.
2. *Session Key Agreement* Message confidentiality is one of the major concerns for any security system. Therefore, in authentication protocol, a shared session key should be established between the participants for encrypting future interactions.
3. *No Online TA* TA should not take part in mutual authentication and key agreement process to decrease the communication overhead and achieve resistance against single point of failure.
4. *Identity Anonymity* The real identities of SMs should not be revealed to any potential attacker even if it is able to intercept communication messages during the authentication and key agreement phase.
5. *Traceability and Revocation* If any malicious activity is detected, the protocol should be able to trace the malicious node and then revoke it.
6. *Perfect Forward Secrecy* To provide security to previous communications, an attacker should not be able to extract the session key of past sessions even if the attacker got the secret parameters of participants.
7. *Distributed Data Storage and Access* A good blockchain-based authentication scheme should support distributed data storage managed by multiple parties rather than a single trusted authority. It should also ensure distributed database access with no malicious modifications.
8. *Resistance to Various Attacks* A perfectly secure system should resist attacks such as modification attacks, replay attacks, man-in-the-middle attacks, and forgery attacks.

4 Proposed scheme

4.1 Initial system setup

The system defines an elliptic curve $E : y^2 = x^3 + \tilde{a}x + \tilde{b}modp$ over a finite field F_p, where $\tilde{a}, \tilde{b} \in \mathbb{Z}_q^*$ and $4\tilde{a}^3 + 27\tilde{b}^2 \neq 0$. An additive cyclic group G is chosen by system on elliptic curve E with generator P and prime order q.

1. Firstly, TA randomly chooses $s \in \mathbb{Z}_q^*$ as private key and calculates $Pb_{TA} = s \cdot P$ as public key.
2. Further TA picks following one way hash functions: $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $h_2 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $h_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $h_4 : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $h_5 : \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $h_6 : \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $h_7 : \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.
3. Cloud Servers are not separately registered as they are already configured during blockchain network formation. However, they announce their public key as $Pb_{CS} = v.P$, where $v \in \mathbb{Z}_q^*$ is chosen randomly. In addition, cloud servers can perform operations on the ledger by invoking transaction processing functions of smart contract in algorithm 1.
4. Finally, TA publishes system public parameters as $par = \{\mathbb{G}, P, q, Pb_{TA}, h_i\}$ and keeps s secret.

Algorithm 1: Transaction processing functions of smart contract at cloud server

```
Input: Invoked transaction's parameter.
/* @param {insertTransactionM} addM */
% Defining the transaction parameter.
% Invoked to insert a new smart meter.
function addMeter(addM):
    let asset = await getAssetRegistry(meterDB);
    let device = await newResource('meterDB', addM.newIDM);
    device.HPBM = addM.newHPBM;
    device.DI = addM.DI;
    return asset.add(device);
end
/* @param {insertTransaction} addf */
% Defining the transaction parameter.
% Invoked to insert a fog node.
function addFogNode(addf):
    let asset = await getAssetRegistry(fogDB);
    let device = await newResource('fogDB', addf.newIDF);
    device.HPBF = addf.newHPBF;
    return asset.add(device);
end
/* @param {removeTransactionM} removeM */
% Invoked to remove a smart meter.
function removeMeter(removeM):
    let assetRegistry = await getAssetRegistry(meterDB);
    let result = await query('selectMeterByIdentity', removeM.IDM);
    await assetRegistry.remove(result);
end
/* @param {removeTransaction} removeF */
% Invoked to remove a fog node.
function removeFogNode(removeF):
    let assetRegistry = await getAssetRegistry(fogDB);
    let result = await query('selectFognodeByIdentity', removeF.IDF);
    await assetRegistry.remove(result);
end
/* @param {updateTransaction} updateF */
% Invoked to update fog node public key.
function updateFogNode(updateF):
    let assetRegistry = await getAssetRegistry(fogDB);
    let device = await newResource('fogDB', updateF.IDF);
    device.HPBF = updateF.newHPBF;
    await assetRegistry.update(device);
end
/* @param {updateTransactionM} updateM */
% Invoked to update smart meter public key.
function updateMeter(updateM):
    let assetRegistry = await getAssetRegistry(meterDB);
    let device = await newResource('meterDB', updateM.IDM);
    device.HPBM = updateM.newHPBM;
    await assetRegistry.update(device);
end
```

Notations: IDM- ID_{m_i}, HPBM- $h_1(Pb_{m_i})$, DI- D_i, IDF- ID_{f_j}, HPBF- $h_1(Pb_{f_j})$

4.2 Blockchain initialization

The TA initiates the fabric network by forming a single organization fabric network, where a cloud server acts as a single peer node and orderer and TA as an administrator. Furthermore, the administrator creates a channel F1 and adds fog node peers to that channel, and later creates channel 1 to add cloud server peers. A cloud server peer is a part of both channel 1 and channel F1 and holds the chaincode (Algorithm 1) of both the channels, while fog node peer keeps chaincode of channel F1. Access rights are defined according to the needs of client applications. Any changes to the ledger are made after the mutual consensus of participating peers to maintain the ledger integrity. Clients such as TA and smart meter can interact with the blockchain and perform operations through API that invokes chaincode functions to perform the transactions.

4.3 Registration phase

This phase involves meter and fog registration with TA. It has following steps.

4.3.1 Fog node registration

1. FN_j picks a random number $t \in \mathbb{Z}_q^*$ and calculates public key $Pb_{f_j} = t \cdot P$. Then FN_j sends registration request $\{Pb_{f_j}, ID_{f_j}\}$ to TA.
2. After receiving the message, TA calculates $PID_{f_j} = h_2(ID_{f_j}, s \cdot Pb_{CS})$, adds $\{h_1(Pb_{f_j}), ID_{f_j}\}$ into cloudchain and sends PID_{f_j} to FN_j.
3. FN_j stores PID_{f_j} securely and ends the registration.

4.3.2 Meter registration

1. Initially, SM_i selects a random number $u \in \mathbb{Z}_q^*$ and calculates public key $Pb_{m_i} = u \cdot P$. Then SM_i sends registration request message $\{ID_{m_i}, Pb_{m_i}\}$ to TA.
2. After receiving the message, TA chooses $d_i \in \mathbb{Z}_q^*$ randomly and calculates $D_i = d_i \cdot P$, $PID_{m_i} = h_2(ID_{m_i}, s \cdot Pb_{CS})$, $S_i = d_i + s \cdot h_1(PID_{m_i} \parallel h_1(Pb_{m_i}))$.Then TA adds $\{h_1(Pb_{m_i}), ID_{m_i}, D_i\}$ in to the cloudchain ledger database through application and returns $\{PID_{m_i}, S_i\}$ to SM via a secure channel.
3. After receiving, SM_i adds $\{h_1(Pb_{m_i}), PID_{m_i}\}$ in to fogchain by interacting with the nearest fog node and keeps S_i secret.

4.4 Mutual authentication and key agreement phase

In this phase, SM, FN and CS mutually authenticate each other and establish a common session key. Furthermore, FN and CS verify SM credentials through their corresponding blockchain ledger.

1. SM_i initiates mutual authentication and key agreement phase with a nearby fog node. Initially, it chooses a random number $r \in \mathbb{Z}_q^*$ and calculates $a = h_1(r \parallel S_i)$, $\mathcal{A} = a.P$, $\overline{S_i} = h_1(S_i.P \parallel ID_{m_i})$, $IP_i = \overline{S_i} \oplus h_1(a.Pb_{CS} \parallel PID_{m_i} \parallel T_{m_i})$, $\kappa = h_3(h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, PID_{m_i})$.Then SM_i sends $M_1 = \{h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, \kappa\}$ to nearby fog node FN_j.
2. FN_j first compares the received timestamp with current timestamp i.e., $|T_m^* - T_m| \leq \Delta T$. If this inequality satisfies then FN_j goes for further calculation. FN_j queries the ledger based on received $h_1(Pb_{m_i})$ and extracts the corresponding PID_{m_i}. Furthermore, FN_j verifies if $\kappa^* = h_3(h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_{m_i}, PID_{m_i})$ is same as received κ. If yes then FN_j selects $b \in \mathbb{Z}_q^*$ randomly and computes $\mathcal{B} = b \cdot P$, $\tilde{B} = b \cdot Pb_{CS}$, , $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A}) \cdot P$, $K_f = (b + t) \cdot l_a$, $\tau = h_4(h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, \tilde{B}, T_m, T_f, K_f, PID_{f_j})$. Lastly, FN_j transmits $M_2 = \{h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, T_m, T_f, K_f, \tau\}$ to CS.
3. After receiving the message, CS firstly verifies both the timestamp i.e, $|T_m^* - T_m| \leq \Delta T$, $|T_f^* - T_f| \leq \Delta T$. If both are valid, then CS queries ledger from cloudchain using received $h_1(Pb_{m_i})$ and $h_1(Pb_{f_j})$ to extract $\{ID_{m_i}, D_i\}$ and ID_{f_j}, respectively. Next, CS performs following steps:
   (a) CS calculates $PID_{m_i} = h_2(ID_{m_i}, v.Pb_{TA})$, $PID_{f_j} = h_2(ID_{f_j}, v.Pb_{TA})$, $\tilde{B} = v.\mathcal{B}$, $\tau^* = h_4(h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, \tilde{B}, T_m, T_f, K_f, PID_{f_j})$,if it matches with received τ then CS goes for further steps.
   (b) CS extracts $\overline{S_i} = IP_i \oplus h_1(v.\mathcal{A} \parallel PID_{m_i} \parallel T_{m_i})$ from the received message and calculates $S_i^* \cdot P = D_i + Pb_{TA}.h_1(PID_{m_i} \parallel h_1(Pb_{m_i}))$. If $h_1(S_i^* \cdot P \parallel ID_{m_i}) = \overline{S_i}$ then SM is valid, otherwise CS stops the process.
   (c) CS randomly generates $c \in \mathbb{Z}_q^*$ and calculates $\mathcal{C} = c \cdot P$, $CP_{CS} = K_f.(c + v)$, $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A}).P$,$l_b = Pb_{f_j} + \mathcal{B}$,$\overline{l_a} = = (c + v).l_a$, $\overline{l_b} = (c + v).l_b$, $SK_{CS} = h_7(CP_{CS}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, $\mu_1 = h_5(\overline{l_a}, PID_{f_j}, T_{CS}, SK_{CS})$, $\mu_2 = h_6(PID_{m_i}, \overline{l_b}, S_i, PID_{f_j}, T_{CS}, SK_{CS})$.
   (d) CS sends $M_3 = \{\overline{l_a}, \overline{l_b}, \mu_1, \mu_2, T_{CS}\}$ to FN_j in the last step.
4. When FN_j receives the message, it first verifies the validity of timestamp, and then it goes for further calculation: $CP_{FN} = (b + t).\overline{l_a}$, $SK_{FN} = h_7(CP_{FN}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, $E_j = PID_{f_j} \oplus PID_{m_i}$. Later FN_j verifies whether $h_5(\overline{l_a}, PID_{f_j}, T_{CS}, SK_{FN}) = \mu_1$ holds or not. If not, fog node aborts the process. At last, FN_j transmits $M_4 = \{\overline{l_b}, E_j, \mu_2, T_f^2, T_{CS}\}$ to SM_i.
5. SM_i first checks the validity of timestamp T_f^2 and then extracts $PID_{f_j} = E_j \oplus PID_{m_i}$. Next, SM_i calculates $CP_{SM} = (a + h_1(PID_{m_i} \parallel \mathcal{A})).\overline{l_b}$, $SK_{SM} = h_7(CP_{SM}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$ and finally accepts SK_{SM} as session key if $h_6(PID_{m_i}, l_b, S_i, PID_{f_j}, T_{CS}, SK_{SM}) = \mu_2$ holds.

At the end of the authentication process, SM, FN, and CS establish a shared session key, which they use for future communications.

5 Security analysis and proof

This section involves the security analysis and proof of the proposed scheme. Firstly, a security model based on the RoR model is defined, and then the proposed BAIoG protocol is proved secure according to the defined security model [6, 43]. At last, informal analysis is described to prove the defense against different security attacks (Tables 3 and 4).

5.1 Security model

In the next step, different components associated with the RoR model are described (Fig. 2).

- ***Participants***: The proposed BAIoG protocol involves three participants that are smart meters, fog nodes, and cloud servers. Let Γ_i^M, Γ_j^{FN}, and Γ_k^{CS} represent the ith instance of smart meter, jth instance of fog node and kth instance of cloud server, respectively which are called oracles.
- ***Partnering***: The instances Γ_i^{Λ} and $\Gamma_j^{\overline{\Lambda}}$ are considered partners if they follow below conditions:
  - Both instances Γ_i^{Λ} and $\Gamma_j^{\overline{\Lambda}}$ are in the accepted state.
  - They establish a common session key and mutually authenticate each other.
  - They both are mutual partners of each other.
- ***Freshness***: The instances Γ_i^{Λ} and $\Gamma_j^{\overline{\Lambda}}$ are considered fresh if their common session key is not compromised or revealed to any adversary $\mathscr{A}$.
- ***Accepted State***: If an instance Γ_i^{Λ} receives the last message according to the actual protocol and goes to accept state, it is in the "accepted state". Moreover, when all the sent and received messages are rearranged in sequence, a session identifier *SID* is established for the current session.

It is assumed that the adversary may intercept, modify or inject new messages during communication between two parties. $\mathscr{A}$ can ask the following oracle queries.

1. $Send(\Gamma_i^{\Lambda}, M)$: When challenger $\mathscr{C}$ receives this query with a message M, it responds according to the real protocol execution steps. This query models an active attack.
2. $Execute(\Gamma_i^U, \Gamma_j^{FN}, \Gamma_k^{CS})$: Adversary $\mathscr{A}$ uses this query to simulate the passive eavesdropping. On receiving this query, $\mathscr{C}$ returns all the exchanged messages between instances Γ_i^M, Γ_j^{FN} and Γ_k^{CS} during protocol execution.

Table 3 Notations used in the proposed scheme

Symbol	Description
E	Elliptic curve over F_p
P	Base point on E
G	Additive cyclic group on curve E
q	Prime order
$\tilde{a}, \tilde{b}$	Elliptic curve coefficients
$h_i(i = 1...7)$	Hash functions
SM_i	ith smart meter
FN_j	jth fog node
CS	Cloud server
$Pb_{TA}, Pb_{m_i}, Pb_{f_j}, Pb_{CS}$	Public keys of TA, SM_i, FN_j and CS, respectively
s, t, u, v	Private keys of TA, FN_j, SM_i and CS, respectively
ID_{m_i}, ID_{f_j}	Identities of smart meter and fog node
PID_{m_i}, PID_{f_j}	Pseudo identities of smart meter and fog node
r, b, c	Session specific random numbers chosen by smart meter, fog node, and cloud server, respectively
SK_{CS}, SK_{FN}, SK_{SM}	Common session key calculated at each side
$\kappa, \tau, \mu_1, \mu_2$	Authentication tokens
T_m, T_f, T_{CS}	Timestamps
$\mathscr{C}$, $\mathscr{A}$	Challenger and adversary
Γ_i^M, Γ_j^{FN}, Γ_k^{CS}	Represent instances of smart meter, fog node and cloud server
q_{h_i}, q_{se}, q_{ex}	Number of hash queries, send queries, and execute queries, respectively.
$Adv_{\mathscr{A}}^{BAIoG}(t)$	Adversary's advantage of breaking BAIoG protocol
$Adv_{\mathscr{A}}^{ECDDH}(t)$	Adversary's advantage of breaking ECDDH problem

Table 4 The simulation of oracles

Hash queries are simulated by maintaining a list L_{h_i}. Whenever $\mathscr{A}$ asks a hash query, it is returned the corresponding entry if it exists in L_{h_i}. Otherwise, a random value from $\{0,1\}^*$ is chosen and returned to $\mathscr{A}$
On receiving $Send(\Gamma_i^M,(FN,START))$ query, $\mathscr{C}$ chooses a random number $r \in \mathbb{Z}_q^*$ and calculates $a = h_1(r \parallel S_i)$, $\mathcal{A} = a.P$, $\overline{S_i} = h_1(S_i.P \parallel ID_{m_i})$, $IP_i = \overline{S_i} \oplus h_1(a.Pb_{CS} \parallel PID_{m_i} \parallel T_{m_i})$, $\kappa = h_3(h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, PID_{m_i})$, and sends $M_1 = \{h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, \kappa\}$ to $\mathscr{A}$
On receiving $Send(\Gamma_j^{FN}, M_1)$ query, $\mathscr{C}$ verifies κ by using previously generated message. Next, $\mathscr{C}$ chooses a random number $b \in \mathbb{Z}_q^*$ and calculates $\mathcal{B} = b.P$, $\tilde{B} = b.Pb_{CS}$, $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A}).P$, $K_f = (b+t).l_a$, $\tau = h_4(h(Pb_{m_i}), h(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, \tilde{B}, T_m, T_f, K_f, PID_{f_j})$, and sends $M_2 = \{h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, T_m, T_f, K_f, \tau\}$ to $\mathscr{A}$.
On receiving $Send(\Gamma_k^{CS}, M_2)$ query, $\mathscr{C}$ verifies τ and IP_i by using messages generated in $Send(\Gamma_i^M,(FN,START))$ and $Send(\Gamma_j^{FN}, M_1)$ query. If both of them are correct, $\mathscr{C}$ chooses a random number $c \in \mathbb{Z}_q^*$ and calculates $\mathcal{C} = c.P$, $CP_{CS} = K_f.(c+v)$, $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A})$, $l_b = Pb_{f_j} + \mathcal{B}$, $\overline{l_a} = (c+v).l_a$, $\overline{l_b} = (c+v).l_b$. Furthermore, $\mathscr{C}$ computes session key at CS side as $SK_{CS} = h_7(CP_{CS}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, and authentication codes $\mu_1 = h_5(\overline{l_a}, PID_{f_j}, T_{CS}, SK_{CS})$, $\mu_2 = h_6(PID_{m_i}, \overline{l_b}, S_i, PID_{f_j}, T_{CS}, SK_{CS})$, and sends $M_3 = \{\overline{l_a}, \overline{l_b}, \mu_1, \mu_2, T_{CS}\}$ to $\mathscr{A}$. Otherwise, $\mathscr{C}$ will reject $\mathscr{A}$'s query and return $\perp$.
For $Send(\Gamma_j^{FN}, M_3)$ query, $\mathscr{C}$ computes $CP_{FN} = (b+t).\overline{l_a}$, $SK_{FN} = h_7(CP_{FN}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, $E_j = PID_{f_j} \oplus PID_{m_i}$ and verifies the correctness of μ_1. $\mathscr{C}$ sends $M_4 = \{\overline{l_b}, E_j, \mu_2, T_f^2, T_{CS}\}$.
For $Send(\Gamma_i^M, M_4)$ query, $\mathscr{C}$ computes $CP_{SM} = (a + h_1(PID_{m_i} \parallel \mathcal{A})).\overline{l_b}$, $SK_{SM} = h_7(CP_{SM}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$ and checks the correctness of μ_2. The $\mathscr{C}$ accepts and terminates the instance Γ_i^M. At last, (M_1, M_2, M_3, M_4) are added into the list L_A
On receiving $Corrupt(\Gamma_i^M)$, $\mathscr{C}$ sends long term secrets $\{PID_{m_i}, S_i\}$ to $\mathscr{A}$.
On receiving $Execute(\Gamma_i^U, \Gamma_j^{FN}, \Gamma_k^{CS})$, $\mathscr{C}$ retrieves (M_1, M_2, M_3, M_4) from list L_A and sends them to $\mathscr{A}$.
For $Reveal(\Gamma_i^\Lambda)$, $\mathscr{C}$ returns the session key if Γ_i^Λ is in accepted state, else outputs $\perp$.
For $Test(\Gamma_i^\Lambda)$, $\mathscr{C}$ selects a random bit $q \in \{0,1\}$ and sends session key if $q = 1$, else sends a random value of similar length to session key.

3. *Reveal*(Γ_i^Λ): When adversary $\mathscr{A}$ performs this query, $\mathscr{C}$ returns the session key if Γ_i^Λ is in accepted state. Otherwise, $\mathscr{C}$ outputs $\perp$.
4. *Corrupt*(Γ_i^Λ): This query helps $\mathscr{A}$ to acquire the long-term secrets of the instance Γ_i^Λ and is used in simulating perfect forward secrecy.
5. *test*(Γ_i^Λ): This query simulates the semantic security of the session key and is performed only once. On receiving this query, $\mathscr{C}$ flips an unbiased coin and gets a bit $q \in \{0,1\}$. If $q = 1$, then the real session key is returned to the adversary. Otherwise, $\mathscr{C}$ returns a random number of similar length to the session key.

5.2 Formal security proof

Theorem I Assume that a probabilistic polynomial-time adversary $\mathscr{A}$ is running against the proposed BAIoG protocol, which can solve the ECDDH (Elliptic Curve Decisional Difine–Hellman) problem with advantage $Adv_{\mathscr{A}}^{ECDDH}(t)$ in time t. $\mathscr{A}$ can make $q_{h_i}(i = 1, 2, ...7)$ hash queries, q_{se} send queries, q_{ex} execute queries. Therefore, $\mathscr{A}$'s advantage of attacking BAIoG protocol is evaluated as:

$$Adv_{\mathscr{A}}^{BAIoG}(t) \le \frac{Q_h}{2^l} + \frac{(q_{se}+q_{ex})^2}{q} + \frac{q_{se}^2}{2^{l-1}} + 2\left|\frac{q_{h_1}+q_{h_4}+q_{h_7}}{q}\right| + 2q_{se}.Adv_{\mathscr{A}}^{ECDDH}(t),$$

Where l is length of hash outputs and $Q_h = \sum_{i=1}^{6} q_{h_i}^2$.

Proof For a given ECDDH problem instance $(P, X = x.P, Y = y.P, Z = z.P)$, $\mathscr{C}$ uses $\mathscr{A}'s$ ECDDH problem solving capability to determine if $Z = x \cdot y \cdot P$ holds. Initially, the challenger $\mathscr{C}$ selects $s \in \mathbb{Z}_q^*$ randomly, and calculates $Pb_{TA} = s \cdot P$. Then $\mathscr{C}$ declares public parameters as $par = \{\mathbb{G}, P, q, Pb_{TA}, h_i\}$. To prove the security of the proposed scheme against adversary $\mathscr{A}$, the sequence of games from G_1 to G_4 is designed. Let $Succ(\mathscr{A})^{G_j}$ denote the event in which $\mathscr{A}$ can successfully guess the value q in the game G_j with the winning advantage of $Pr[Succ(\mathscr{A})^{G_j}]$.

Game G_0: In the initial game G_0, the adversary $\mathscr{A}$ performs the actual attack simulation in the ROR model against the proposed BAIoG protocol. Therefore, in accordance with the semantic security definition [44], we have

Smart Meter (SM_i)	Fog Node (FN_j)	Cloud Server (CS)
Step 1: Chooses $r \in \mathbb{Z}_q^*$, $a = h_1(r \parallel S_i)$, $\mathcal{A} = a.P$, $\overline{S_i} = h_1(S_i.P \parallel ID_{m_i})$, $IP_i = \overline{S_i} \oplus h_1(a.Pb_{CS} \parallel PID_{m_i} \parallel T_{m_i})$, $\kappa = h_3(h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, PID_{m_i})$. $\xrightarrow{M_1=\{h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, \kappa\}}$	Step 2: Checks $\lvert T_m^* - T_m\rvert \le \Delta T$, queries ledger based on $h_1(Pb_{m_i})$ to extract PID_{m_i}, verifies $\kappa^* = h_3(h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_{m_i}, PID_{m_i}) \stackrel{?}{=} \kappa$, $b \in Z_q^*$, $\mathcal{B} = b.P$, $\tilde{B} = b.Pb_{CS}$, $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A}).P$, $K_f = (b+t).l_a$, $\tau = h_4(h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, \tilde{B}, T_m, T_f, K_f, PID_{f_j})$. $\xrightarrow{M_2=\{h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, T_m, T_f, K_f, \tau\}}$	Step 3: Verifies $\lvert T_m^* - T_m\rvert \le \Delta T$, $\lvert T_f^* - T_f\rvert \le \Delta T$, queries ledger based on $h_1(Pb_{m_i})$ and $h_1(Pb_{f_j})$ to extract $\{ID_{m_i}, D_i\}$ and ID_{f_j}. (a) $PID_{m_i} = h_2(ID_{m_i}, v.Pb_{TA})$, $PID_{f_j} = h_2(ID_{f_j}, v.Pb_{TA})$, $\tilde{B} = v.B$, verifies $\tau^* = h_4(h(Pb_{m_i}), h(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, \tilde{B}, T_m, T_f, K_f, PID_{f_j}) \stackrel{?}{=} \tau$. (b) $\overline{S_i} = IP_i \oplus h_1(v.\mathcal{A} \parallel PID_{m_i} \parallel T_{m_i})$, $S_i^*.P = D_i + Pb_{TA}.h_1(PID_{m_i} \parallel h(Pb_{m_i}))$. If $h_1(S_i^*.P \parallel ID_{m_i}) = \overline{S_i}$ then SM is valid. (c) $c \in Z_q^*$, $\mathcal{C} = c.P$, $CP_{CS} = K_f.(c+v)$, $l_a = \mathcal{A} + h_1(PID_{m_i} \parallel \mathcal{A}).P$, $l_b = Pb_{f_j} + \mathcal{B}$, $\overline{l_a} = (c+v).l_a$, $\overline{l_b} = (c+v).l_b$, $SK_{CS} = h_7(CP_{CS}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, $\mu_1 = h_5(\overline{l_a}, PID_{f_j}, T_{CS}, SK_{CS})$, $\mu_2 = h_6(PID_{m_i}, \overline{l_b}, S_i, PID_{f_j}, T_{CS}, SK_{CS})$. $\xleftarrow{M_3=\{\overline{l_a}, \overline{l_b}, \mu_1, \mu_2, T_{CS}\}}$
Step 5: Verifies timestamp then computes $PID_{f_j} = E_j \oplus PID_{m_i}$, $CP_{SM} = (a + h_1(PID_{m_i} \parallel \mathcal{A})).\overline{l_b}$, $SK_{SM} = h_7(CP_{SM}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, verifies if $h_6(PID_{m_i}, \overline{l_b}, S_i, PID_{f_j}, T_{CS}, SK_{SM}) = \mu_2$ holds.	Step 4: Verifies timestamp then $CP_{FN} = (b+t).\overline{l_a}$, $SK_{FN} = h_7(CP_{FN}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, $E_j = PID_{f_j} \oplus PID_{m_i}$. FN_j verifies $h_5(\overline{l_a}, PID_{f_j}, T_{CS}, SK_{FN}) = \mu_1$ holds or not.	

Fig. 2 Mutual authentication and key agreement

$$Adv_{\mathscr{A}}^{BAIoG}(t) = |2.Pr[Succ(\mathscr{A})^{G_0}] - 1|. \quad (1)$$

Game G_1: In this game, hash oracle queries $h_i(m)(i = 1, 2, ...7)$ are simulated. $\mathscr{C}$ maintains a list l_{h_i} for each corresponding hash oracle h_i. When $\mathscr{C}$ receives $\mathscr{A}$'s query for $h_i(m)$, $\mathscr{C}$ looks up the corresponding entry (m, v) in list l_{h_i}; if it exists, $\mathscr{C}$ returns the v to $\mathscr{A}$. Otherwise, $\mathscr{C}$ selects $v \in \mathbb{Z}_q^*$ randomly and sends v to $\mathscr{A}$. Furthermore, $\mathscr{C}$ stores v

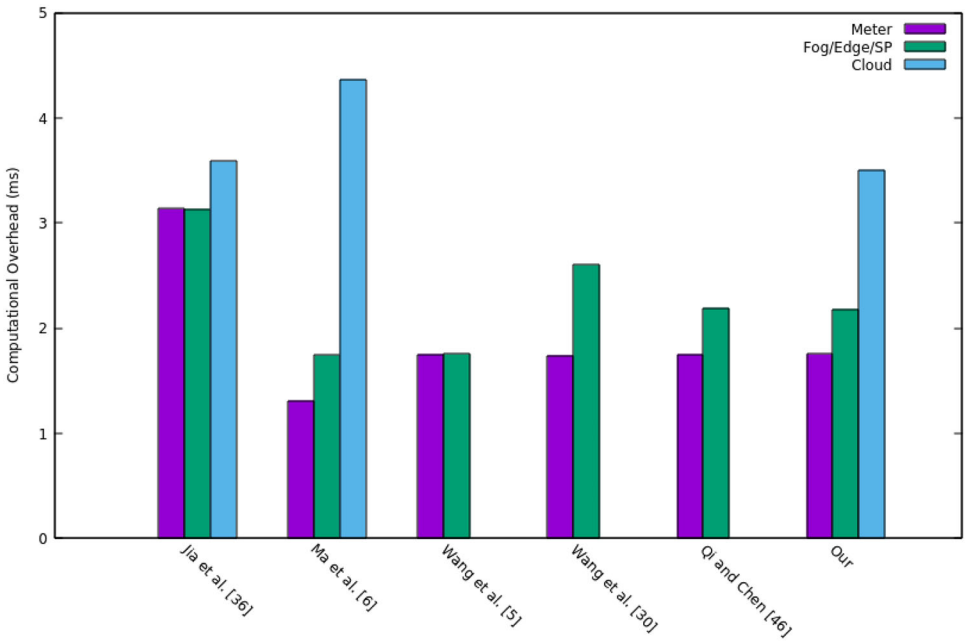

Fig. 3 Cryptographic operation's computational time comparison

to the list l_{h_i} for future queries. Other queries such as send, execute are simulated by $\mathscr{C}$ in similar way as actual attack and the corresponding response is returned to $\mathscr{A}$. It can be observed that this game G_1 is completely indistinguishable from G_0. Consequently,

$$Pr[Succ(\mathscr{A})^{G_1}] = Pr[Succ(\mathscr{A})^{G_0}]. \tag{2}$$

Game G_2: This game simulates all oracle queries similar to game G_1, however, if there is a collision on the hash h_i output or the transcripts, then G_2 halts. Thus, according to birthday paradox, it follows that

$$|Pr[Succ(\mathscr{A})^{G_2}] - Pr[Succ(\mathscr{A})^{G_1}]| \leq \frac{Q_h}{2^{l+1}} + \frac{(q_{se}+q_{ex})^2}{2q}, \tag{3}$$

where $Q_h = \sum_{i=1}^{6} q_{h_i}^2$.

Game G_3: This game simulates all oracle queries similar to game G_2. However, G_3 aborts if $\mathscr{A}$ is able to generate fake $(\kappa, \tau, \mu_1, \mu_2)$ without asking random oracle query. Thus [45],

$$|Pr[Succ(\mathscr{A})^{G_3}] - Pr[Succ(\mathscr{A})^{G_2}]| \leq \frac{q_{se}^2}{2^l}. \tag{4}$$

Game G_4: In this game, send queries are modified as well as adversary $\mathscr{A}$ is permitted to query $Corrupt(\Gamma_i^M)$ to extract long term secrets such as $\{PID_{m_i}, S_i\}$. When $\mathscr{C}$ receives $\mathscr{A}$'s *Send* queries, $\mathscr{C}$ selects a random instance $(\Gamma_i^M, \Gamma_j^{FN}, \Gamma_k^{CS})$ and answers as following:

1. After receiving $\mathscr{A}$'s $Send(\Gamma_i^M, (FN, START))$ query, $\mathscr{C}$ computes $\mathcal{A} = \mathcal{A}_0$ and creates values $\overline{S_i}$, IP_i, κ, T_m in similar way as game G_3. Lastly, $\mathscr{C}$ transmits $M_1 = \{h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, \kappa\}$ to $\mathscr{A}$.
2. After receiving $\mathscr{A}$'s $Send(\Gamma_j^{FN}, M_1)$ query, $\mathscr{C}$ computes $\mathcal{B} = \mathcal{B}_0$, $\tilde{B} = v.\mathcal{B}$, $l_a = X$, $K_f = Z$, and creates τ, T_f in similar way as game G_3. Lastly, $\mathscr{C}$ transmits $M_2 = \{h_1(Pb_{m_i}), h_1(Pb_{f_j}), IP_i, \mathcal{A}, \mathcal{B}, T_m, T_f, K_f, \tau\}$ to $\mathscr{A}$.
3. After receiving $\mathscr{A}$'s $Send(\Gamma_k^{CS}, M_2)$ query, $\mathscr{C}$ selects a $c \in \mathbb{Z}_q^*$ randomly and computes $\mathcal{C} = c.P$, $CP_{CS} = (c+v).K_f$, $l_a = X$, $l_b = Y$, $\overline{l_a} = (c+v).l_a$, $\overline{l_b} = (c+v).l_b$. Furthermore, $\mathscr{C}$ creates values PID_{m_i}, PID_{f_j}, μ_1, μ_2, T_{CS} in similar way as game G_3 and computes $SK_{CS} = h_7(CP_{CS}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$. Lastly, $\mathscr{C}$ adds $(c, \mathcal{C})$ in a list L_c and transmits $M_3 = \{\overline{l_a}, \overline{l_b}, \mu_1, \mu_2, T_{CS}\}$ to $\mathscr{A}$.
4. After receiving $\mathscr{A}$'s $Send(\Gamma_j^{FN}, M_3)$ query, $\mathscr{C}$ searches $(c, \mathcal{C})$ in the list L_c and computes $CP_{FN} = (c+v).K_f$, $SK_{FN} = h_7(CP_{FN}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$. Lastly, $\mathscr{C}$ transmits $M_4 = \{\overline{l_b}, E_j, \mu_2, T_f^2, T_{CS}\}$ to $\mathscr{A}$.
5. After receiving $\mathscr{A}$'s $Send(\Gamma_i^M, M_4)$ query, $\mathscr{C}$ searches $(c, \mathcal{C})$ in the list L_c, computes $CP_{SM} = (c+v).Z$, $SK_{SM} = h_7(CP_{SM}, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$ and terminates the instance.

Let's assume a differentiator $\mathscr{D}$ exists that is capable of successfully distinguishing game G_4 from G_3. This differentiator $\mathscr{D}$ can be used by challenger $\mathscr{C}$ in solving the ECDDH problem. In the description above, $\mathscr{C}$ simulates all the queries with no knowledge about a, b, c. If $(c+v).Z = (c+v).xyP$, i.e., $Z = xyP$ holds, $\mathscr{D}$ follows game G_3, and $\mathscr{C}$ returns 1. Otherwise, $\mathscr{D}$ follows game G_4,

and $\mathscr{C}$ returns 0. $\mathscr{D}$ picks an instance with the probability $\frac{1}{q_{se}}$. Therefore,

$$|Pr[Succ(\mathscr{A})^{G_4}] - Pr[Succ(\mathscr{A})^{G_3}]| \leq q_{se}.Adv_{\mathscr{A}}^{ECDDH}(t). \tag{5}$$

In the current game, $CP_M = CP_{FN} = CP_{CS} = (c + v).Z$ is a random number and not dependent on x and y. The occurrence of the following events is a must for $\mathscr{A}$ to differentiate a true SK and a random number.

E_1: $\mathscr{A}$ has successfully impersonated an SM and forged $M_1 = \{h_1(Pb_{m_i}), IP_i, \mathcal{A}, T_m, \kappa\}$. To forge M_1 successfully, $\mathscr{A}$ should correctly generate IP_i. $\mathscr{A}$ is permitted to query $Corrupt(\Gamma_j^M)$ to extract the long term secrets. However, calculating IP_i is still difficult because a_0 is an unknown number. Therefore, we have

$$Pr[E_1] \leq \frac{q_{h_1}}{q}.$$

E_2 $\mathscr{A}$ has also impersonated FN and forged M_2. To forge M_2 successfully, $\mathscr{A}$ needs to correctly calculate the value of τ. However, the value of τ cannot be calculated without b_0, which is an unknown number. Therefore, we have

$$Pr[E_2] \leq \frac{q_{h_4}}{q}.$$

E_3 $\mathscr{A}$ queries h_7 hash oracle for input $\{(c + v).Z, PID_{m_i}, PID_{f_j}, l_a, T_{CS}\}$. Thus,

$$Pr[E_3] \leq \frac{q_{h_7}}{q},$$

and therefore

$$Pr[Succ(\mathscr{A})^{G_4}] = \frac{1}{2} + \frac{q_{h_1} + q_{h_4} + q_{h_7}}{q}. \tag{6}$$

On solving equations (1)-(6), following result is obtained

$$Adv_{\mathscr{A}}^{BAIoG}(t) \leq \frac{Q_h}{2^l} + \frac{(q_{se} + q_{ex})^2}{q} + \frac{q_{se}^2}{2^{l-1}} + 2\left|\frac{q_{h_1} + q_{h_4} + q_{h_7}}{q}\right| + 2q_{se}.Adv_{\mathscr{A}}^{ECDDH}(t).$$

5.3 Analysis of security goals

Next, we discuss about the security goals achieved by the proposed scheme.

1. *Mutual Authentication CS* authenticates SM_i and FN_j by verifying the validation of IP_i and τ, respectively. Similarly, SM_i verifies both CS and FN_j by checking if μ_2 is valid. FN_j verifies CS and SM_i by checking the validity of μ_1 and κ, respectively. Furthermore, from the proof of theorem I, It is known that no polynomial-time adversary can forge a valid login message. Therefore, the proposed BAIoG protocol achieves mutual authentication.
2. *Session Key Agreement* All the participants calculate a common parameter $CP = (a + h_1(PID_{m_i} \parallel \mathcal{A})).(b + t).(c + v)$ that helps in computing common session key SK, where a, b, and c are generated randomly for present session only. Therefore, the proposed scheme achieves common session key establishment.
3. *No Online TA* In the proposed scheme, TA is not involved during the key agreement and authentication phase. The data is stored on the blockchain, and verification and revocation are done by invoking blockchain transactions.
4. *Identity Anonymity* In the proposed BAIoG protocol, the real identity of an SM_i is included in $\overline{S_i} = h_1(S_i.P \parallel ID_{m_i})$, which is further hidden in $IP = \overline{S_i} \oplus h_1(a.Pb_{CS} \parallel PID_{m_i} \parallel T_{m_i})$. To calculate the real identity of smart meter, attacker needs to solve the ECDL (Elliptic Curve Discrete Logarithm) problem. Therefore, the proposed scheme ensures identity anonymity.
5. *Traceability and Revocation* If any malicious activity is detected, the suspicious SM_i or FN_j will be traced by the cloud server that reports it to TA. Then, TA will invoke remove transaction, and then the details of that node will be deleted from the blockchain.
6. *Perfect Forward Secrecy* The session key in our proposed BAIoG protocol is calculated as $SK = h_7(CP, PID_{m_i}, PID_{f_j}, l_a, T_{CS})$, where CP is associated with session-specific random numbers such as a, b, and c, which are not transmitted on a public channel. Therefore, even if an attacker is able to extract private keys, it cannot compute the session key without solving the ECDH (Elliptic-curve Diffie–Hellman) problem. Thus, the proposed scheme ensures perfect forward secrecy.
7. *Distributed Data Storage and Access* Distributed data storage and access are achieved by deploying a hyperledger fabric network on fog nodes and cloud servers by considering them as fabric peers. The fabric peers participate in the consensus process, and no changes can be made without endorsement from peers. Therefore, the proposed scheme achieves distributed database access and storage.
8. *Stolen Verifier Attack* Since all the verification data is available inside the blockchain ledger and the data can only be accessed by invoking transactions; hence there is no need to maintain a verifier table. Therefore, the proposed scheme is secure against stolen verifier attack.

9. *Impersonation Attack* The impersonation attack is possible only if the attacker can break the mutual authentication security proven in the theorem I.
10. *Replay Attack* Since the messages M_1, M_2, M_3, and M_4 include timestamps T_m, T_f, and T_{CS}, the proposed scheme can verify the freshness of the received message. Therefore, the proposed BAIoG scheme provides security against replay attack.
11. *Man-in-the-middle Attack* The detailed analysis described above proves that the proposed scheme is secure against replay attack, impersonation attack and also achieves authentication and integrity. Therefore, we can assert that the proposed BAIoG protocol is also secure against man-in-the-middle attack.

6 Performance analysis

In this section, we analyze the performance of the proposed BAIoG protocol in terms of communication and computational overhead. Also, we compare the proposed protocol with other existing schemes to prove the computational and communication efficiency.

6.1 Implementation setup

For evaluation of cryptographic operations, we use Crypto++ library[1] and pairing-based cryptography (PBC) library[2] on a laptop with Intel(R) Core(TM) i5-8250U CPU@1.60 GHz and 8 GB RAM running on Ubuntu 18.04 LTS OS. The Cryptographic operations are evaluated in terms of running time which is calculated by taking an average of 10 consecutive executions of each operation with randomly generated inputs. For evaluating ECC operations, We use a non-singular elliptic curve $E : y^2 = x^3 + ax + bmodp$ over a finite field, where p is a 512-bit prime number and $a, b \in \mathbb{Z}_q^*$. In addition, G is an additive group on the elliptic curve E with order q(160 bits prime number) and generator P. Cryptographic operations such as bilinear paring (T_{bp}), ECC scalar multiplication (T_{sm}), general hash function (T_h) and ECC point addition (T_{pa}) takes 2.249 ms, 0.432 ms, 0.004 ms and 0.003 ms, respectively.

We evaluate blockchain operations using hyperledger composer v0.20.9[3] in which a basic blockchain business network is deployed to a single organization fabric network instance running on AWS EC2 instance with Intel Xeon@2.5 GHz, 1 GB RAM, and Ubuntu 18.04 LTS OS. There are three main components in a simple fabric network, peer node, fabric CA and orderer that run inside a docker container. Blockchain operations are performed using a smart contract (chaincode), which is installed on fabric peers and contains logic for remove, insert, update, and query transactions. A REST server is deployed for the underlying business network to calculate the execution times for transactions.

6.2 Computational cost analysis

The computational cost is the running times of various cryptographic operations performed during authentication and key agreement phase of the proposed scheme. In this subsection, the proposed BAIoG protocol is compared with other existing blockchain and fog based key agreement schemes such as Jia et al. [36], Ma et al. [6], Wang et al. [5], Wang et al. [30], Qi and Chen etc. [46], in terms of computational costs. In the proposed scheme, during key agreement and authentication phase four scalar multiplication and seven hash operations are needed at smart meter side. Fog node performs five scalar multiplication, five hash function and one point addition operation. CS needs eight scalar multiplication, three point addition and ten hash function operations. Therefore, total computational cost at smart meter, fog node and cloud server is $4T_{sm} + 7T_h \approx 1.756$, $5T_{sm} + T_{pa} + 5T_h \approx 2.183$, and $8T_{sm} + 3T_{pa} + 10T_h \approx 3.505$, respectively. Based on this, the proposed scheme's comparison is shown in figure 3 and table 5. Our proposed scheme is compared with both two party and three party key agreement schemes. In three party key agreement schemes smart meter, fog node and cloud server are shown whereas for a two party key agreement scheme cloud server is skipped. The proposed BAIoG scheme takes similar computational cost to other schemes at smart meter level except [6]. The scheme [6] is proposed for vehicular network which is a highly mobile and dynamic network that needs very low computational cost. At fog node level our scheme takes lesser time as compared to [30, 36], and [46] and slightly higher than [5, 6]. At cloud server level our scheme takes lesser computational cost than all existing three party key agreement scheme. Since fog nodes and cloud servers have

[1] "Crypto++ Library 8.4 | Free C++ Class Library of Cryptographic Schemes", https://www.cryptopp.com/.

[2] "PBC Library—Pairing-Based Cryptography—About", https://crypto.stanford.edu/pbc/.

[3] "Installing the development environment | Hyperledger Composer", https://hyperledger.github.io/composer/v0.19/installing/development-tools.

Table 5 Comparison of computational time of cryptographic operations

Schemes	Smart meter	Fog node/edge node/service provider	Cloud server
Jia et al. [36]	$2T_{sm} + 6T_h + T_{bp}$	$2T_{sm} + 4T_h + T_{bp}$	$3T_{sm} + 11T_h + T_{bp}$
Ma et al. [6]	$3T_{sm} + 4T_h$	$4T_{SM} + 4T_h$	$10T_{sm} + 11T_h$
Wang et al. [5]	$4T_{sm} + T_{pa} + 5T_h$	$4T_{sm} + T_{pa} + 6T_h$	–
Wang et al. [30]	$4T_{sm} + T_{pa} + 2T_h$	$6T_{sm} + T_{pa} + 2T_h$	–
Qi and Chen [46]	$4T_{sm} + T_{pa} + 4T_h$	$5T_{sm} + 2T_{pa} + 6T_h$	–
Our scheme	$4T_{sm} + 7T_h$	$5T_{sm} + T_{pa} + 5T_h$	$8T_{sm} + 3T_{pa} + 10T_h$

enough computational power, their computational cost does not much affect the overall performance. Therefore, the proposed scheme is efficient in terms of computational cost with the additional features of three-layer blockchain-based architecture and three-party key agreement.

The execution times of hyperledger composer operations are shown in Table 6. We deploy a basic blockchain business network to hyperledger fabric instance for a single organization running on a local system and the AWS cloud instance. The execution times for different transactions are calculated using the REST server generated for the underlying business network on both the local system and AWS cloud.

6.3 Communication cost analysis

In this subsection, we describe the communication cost analysis of the proposed BAIoG scheme. The communication overhead is caused by attachments sent along with the original message, such as pseudo-identity, signature, and other security parameters for secure communication. The sizes of the different parameters such as $\mathbb{G}$, hash function digest, identity, and timestamp are considered 40 bytes, 20 bytes, 20 bytes, and 4 bytes, respectively.

In the proposed scheme, four messages M_1, M_2, M_3 and M_4 are exchanged during authentication and key agreement phase. M_1, M_4 are exchanged between smart meter and fog node, and M_2, M_3 are exchanged between fog node and cloud server. Therefore, the total communication cost between the smart meter and fog node is 192 bytes, and between the fog node and the cloud server is 332 bytes. Table 7 shows the communication cost comparison between the proposed scheme and other existing schemes. Our scheme takes lesser communication cost than other schemes except [5] and [36]. However, it is acceptable because Jia et al.'s [36] scheme is pairing based that takes more computational cost and Wang et al.'s [5] scheme is a two party key agreement scheme with scalability issue. Finally, we conclude that the proposed BAIoG protocol provides better communication and computational efficiency with some extra features of improved blockchain handling scalability and distributed database access and storage.

Table 6 Running times of fabric chaincode operations (in seconds)

Operation	At local system	At cloud
Insert transaction	2.55	2.70
Update transaction	2.44	2.65
Remove transaction	2.61	2.79
Query transaction	0.55	0.65

Table 7 Comparison of communication overhead

Schemes	Communication cost between smart meter and fog node	Communication cost between fog node and cloud server
Jia et al. [36]	188 bytes	252 bytes
Ma et al. [6]	228 bytes	372 bytes
Wang et al. [5]	168 bytes	–
Wang et al. [30]	228 bytes	–
Qi and Chen [46]	208 bytes	–
Our scheme	192 bytes	332 bytes

7 Conclusion

The increasing number of customers and dependency on a single trusted authority make the SG environment vulnerable to many performance and security issues such as single point of failure, higher latency, more computational burden on TA, centralization, etc. Since the inception of fog computing and blockchain, their features have attracted both academia and industry. This paper utilizes blockchain technology and fog computing and creates a three-layer architecture to achieve better scalability and less dependency on a single trusted authority. To provide secure

communication between the smart meter, fog node, and cloud server, we propose a privacy-preserving mutual authentication and key agreement scheme that supports common key establishment between three parties. The formal and informal security analysis shows that the proposed scheme provides session key security and achieves all the security goals. Furthermore, performance analysis and comparison prove the computational and communication efficiency of the proposed scheme.

In future, we will design a data offloading scheme for the fog-based SG environment in which secure data transmission can be achieved through the proposed mutual authentication and key agreement scheme. We will also modify the proposed scheme to be suitable for non-trusted fog node. Furthermore, we will implement the proposed scheme's hyperledger fabric operations on Raspberry Pi to check their feasibility on a real-time smart meter.

Author contributions Conceptualization: [AT]; Methodology: [AT]; Formal analysis and investigation: [AT]; Writing—original draft preparation: [AT]; Writing—review and editing: [ST]; Supervision: [ST].

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

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Fang, D., Qian, Y., Hu, R.Q.: A flexible and efficient authentication and secure data transmission scheme for IoT applications. IEEE Internet Things J. **7**(4), 3474–3484 (2020). https://doi.org/10.1109/JIOT.2020.2970974
2. Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., Markakis, E.K.: A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. IEEE Commun. Surv. Tutor. **22**(2), 1191–1221 (2020). https://doi.org/10.1109/COMST.2019.2962586
3. Mollah, M.B., Zhao, J., Niyato, D., Lam, K.Y., Zhang, X., Ghias, A.M.Y.M., Koh, L.H., Yang, L.: Blockchain for future smart grid: a comprehensive survey. IEEE Internet Things J. **8**(1), 18–43 (2021). https://doi.org/10.1109/JIOT.2020.2993601
4. Garg, S., Kaur, K., Kaddoum, G., Rodrigues, J.J.P.C., Guizani, M.: Secure and lightweight authentication scheme for smart metering infrastructure in smart grid. IEEE Trans. Ind. Inform. **16**(5), 3548–3557 (2020). https://doi.org/10.1109/TII.2019.2944880
5. Wang, J., Wu, L., Choo, K.R., He, D.: Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure. IEEE Trans. Ind. Inform. **16**(3), 1984–1992 (2020). https://doi.org/10.1109/TII.2019.2936278
6. Ma, M., He, D., Wang, H., Kumar, N., Choo, K.R.: An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks. IEEE Internet Things J. **6**(5), 8065–8075 (2019). https://doi.org/10.1109/JIOT.2019.2902840
7. Aazam, M., Zeadally, S., Harras, K.A.: Fog computing architecture, evaluation, and future research directions. IEEE Commun. Mag. **56**(5), 46–52 (2018)
8. Hu, P., Dhelim, S., Ning, H., Qiu, T.: Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
9. Singh, S., Chaurasiya, V.K.: Mutual authentication scheme of IoI devices in fog computing environment. Clust. Comput. **24**, 1643–1657 (2020)
10. Khalid, U., Asim, M., Baker, T., Hung, P.C., Tariq, M.A., Rafferty, L.: A decentralized lightweight blockchain-based authentication mechanism for IoI systems. Clust. Comput. **23**(3), 2067–2087 (2020)
11. Faheem, M., Butt, R.A., Raza, B., Ashraf, M.W., Begum, S., Ngadi, M.A., Gungor, V.C.: Bio-inspired routing protocol for wsn-based smart grid applications in the context of industry 4.0. Trans. Emerg. Telecommun. Technol. **30**(8), e3503 (2019a)
12. Faheem, M., Butt, R.A., Raza, B., Ashraf, M.W., Ngadi, M.A., Gungor, V.C.: A multi-channel distributed routing scheme for smart grid real-time critical event monitoring applications in the perspective of industry 4.0. Int. J. Ad Hoc Ubiquitous Comput. **32**(4), 236–256 (2019b)
13. Guan, Z., Si, G., Zhang, X., Wu, L., Guizani, N., Du, X., Ma, Y.: Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities. IEEE Commun. Mag. **56**(7), 82–88 (2018)
14. Badra, M., Zeadally, S.: Design and performance analysis of a virtual ring architecture for smart grid privacy. IEEE Trans. Inf. Forensics Security **9**(2), 321–329 (2014)
15. Gong, Y., Cai, Y., Guo, Y., Fang, Y.: A privacy-preserving scheme for incentive-based demand response in the smart grid. IEEE Trans. Smart Grid **7**(3), 1304–1313 (2015)
16. Wu, D., Zhou, C.: Fault-tolerant and scalable key management for smart grid. IEEE Trans. Smart Grid **2**(2), 375–381 (2011)
17. Fouda, M.M., Fadlullah, Z.M., Kato, N., Lu, R., Shen, X.S.: A lightweight message authentication scheme for smart grid communications. IEEE Trans. Smart Grid **2**(4), 675–685 (2011)
18. Xia, J., Wang, Y.: Secure key distribution for the smart grid. IEEE Trans. Smart Grid **3**(3), 1437–1443 (2012)
19. Park, J.H., Kim, M., Kwon, D.: Security weakness in the smart grid key distribution scheme proposed by Xia and Wang. IEEE Trans. Smart Grid **4**(3), 1613–1614 (2013)
20. Liu, N., Chen, J., Zhu, L., Zhang, J., He, Y.: A key management scheme for secure communications of advanced metering infrastructure in smart grid. IEEE Trans. Ind. Electron. **60**(10), 4746–4756 (2012)
21. Wan, Z., Wang, G., Yang, Y., Shi, S.: Skm: scalable key management for advanced metering infrastructure in smart grids. IEEE Trans. Ind. Electron. **61**(12), 7055–7066 (2014)
22. Tsai, J.L., Lo, N.W.: Secure anonymous key distribution scheme for smart grid. IEEE Trans. Smart Grid **7**(2), 906–914 (2015)
23. Jo, M., Jangirala, S., Das, A.K., Li, X., Khan, M.K.: Designing anonymous signature-based authenticated key exchange scheme for IoT-enabled smart grid systems. IEEE Trans. Ind. Inform. **17**(7), 4425–4436 (2020)
24. Odelu, V., Das, A.K., Wazid, M., Conti, M.: Provably secure authenticated key agreement scheme for smart grid. IEEE Trans. Smart Grid **9**(3), 1900–1910 (2016)
25. Chen, Y., Martínez, J.F., Castillejo, P., López, L.: An anonymous authentication and key establish scheme for smart grid: Fauth. Energies **10**(9), 1354 (2017)

26. He, D., Wang, H., Khan, M.K., Wang, L.: Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography. IET Commun. **10**(14), 1795–1802 (2016)
27. Abbasinezhad-Mood, D., Nikooghadam, M.: An anonymous ECC-based self-certified key distribution scheme for the smart grid. IEEE Trans. Ind. Electron. **65**(10), 7996–8004 (2018)
28. Zhang, L., Zhao, L., Yin, S., Chi, C.H., Liu, R., Zhang, Y.: A lightweight authentication scheme with privacy protection for smart grid communications. Fut. Gen. Comput. Syst. **100**, 770–778 (2019)
29. Sadhukhan, D., Ray, S., Obaidat, M.S., Dasgupta, M.: A secure and privacy preserving lightweight authentication scheme for smart-grid communication using elliptic curve cryptography. J. Syst. Architect. **114**, 101938 (2021)
30. Wang, W., Huang, H., Zhang, L., Su, C.: Secure and efficient mutual authentication protocol for smart grid under blockchain. Peer-to-Peer Netw. Appl. **14**, 2681–2693 (2020)
31. Deng, L., Gao, R.: Certificateless two-party authenticated key agreement scheme for smart grid. Inf. Sci. **543**, 143–156 (2021)
32. Khan, A.A., Kumar, V., Ahmad, M., Rana, S.: Lakaf: Lightweight authentication and key agreement framework for smart grid network. J. Syst. Architect. **116**, 102053 (2021)
33. Badar, H.M.S., Qadri, S., Shamshad, S., Ayub, M.F., Mahmood, K., Kumar, N.: An identity based authentication protocol for smart grid environment using physical uncloneable function. IEEE Trans. Smart Grid **12**(5), 4426–4434 (2021)
34. Zhang, L., Zhu, Y., Ren, W., Wang, Y., Choo, K.K.R., Xiong, N.N.: An energy efficient authentication scheme based on Chebyshev chaotic map for smart grid environments. IEEE Internet Things J. (2021). https://doi.org/10.1109/JIOT.2021.3078175
35. Chen, C.M., Chen, L., Huang, Y., Kumar, S.: Lightweight authentication protocol in edge-based smart grid environment. EURASIP J. Wirel. Commun. Netw. **1**, 1–18 (2021)
36. Jia, X., He, D., Kumar, N., Choo, K.K.R.: Authenticated key agreement scheme for fog-driven IoT healthcare system. Wirel. Netw. **25**(8), 4737–4750 (2019)
37. Zhu, L., Li, M., Zhang, Z., Xu, C., Zhang, R., Du, X., Guizani, N.: Privacy-preserving authentication and data aggregation for fog-based smart grid. IEEE Commun. Mag. **57**(6), 80–85 (2019)
38. Khan, H.M., Khan, A., Jabeen, F., Rahman, A.U.: Privacy preserving data aggregation with fault tolerance in fog-enabled smart grids. Sustain. Cities Soc. **64**, 102522 (2021)
39. Zhang, H., Wang, J., Ding, Y.: Blockchain-based decentralized and secure keyless signature scheme for smart grid. Energy **180**, 955–967 (2019)
40. Chen, S., Yang, L., Zhao, C., Varadarajan, V., Wang, K.: Double-blockchain assisted secure and anonymous data aggregation for fog-enabled smart grid. Engineering (2020). https://doi.org/10.1016/j.eng.2020.06.018
41. Naseer, O., Ullah, S., Anjum, L.: Blockchain-based decentralized lightweight control access scheme for smart grids. Arab. J. Sci. Eng. **46**, 8233–8243 (2021)
42. Lee, H., Ryu, J., Lee, Y., Won, D.: Security analysis of blockchain-based user authentication for smart grid edge computing infrastructure. In: 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM). IEEE, pp 1–4 (2021)
43. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, pp. 139–155 (2000)
44. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: International Workshop on Public Key Cryptography. Springer, Berlin, pp. 65–84 (2005)
45. Zhang, J., Zhong, H., Cui, J., Xu, Y., Liu, L.: Smaka: secure many-to-many authentication and key agreement scheme for vehicular networks. IEEE Trans. Inf. Forensics Security **16**, 1810–1824 (2020)
46. Qi, M., Chen, J.: Two-pass privacy preserving authenticated key agreement scheme for smart grid. IEEE Syst. J. **15**(3), 3201–3207 (2020)

Ashish Tomar has received B.E. (Information Technology) from SGSITS, Indore, India and M.Tech. from ABV-Indian Institute of Information Technology and Management, Gwalior, India in 2015 and 2018, respectively. At present, he is pursuing Full Time Ph.D. in computer science and engineering at IIT (ISM), Dhanbad, India. His research areas of interest include cryptography authentication, IoT security etc.

Sachin Tripathi Sachin Tripathi received the B.Tech. degree from Chhatrapati Shahu Ji Maharaj University, Kanpur, India. He received the M.Tech. and the Ph.D. degree in computer science and engineering from Indian Institute of Technology (ISM), Dhanbad, India. He is currently working as associate professor with the Indian Institute of Technology (ISM), Dhanbad, India. He is teaching computer science subjects for over more than fourteen years. He has contributed about 100 research papers. He has authored a book titled "Enhancements on Internet Applications: Multicast, Secure E-Mail Messaging and E-Business". He is an Associate Editor of "International Journal of Communication System", Wiley. His research interests mainly focus on group security, cryptography, ad-hoc network, and artificial intelligence.