Chapter 3
Blockchain-Based Fog Computing

Anusha Vangala and Ashok Kumar Das

3.1 Introduction

Fog computing is a distributed computing application consisting of a number of servers that can perform computation, networking and provide storage similar to the servers in a cloud data center. It essentially aims to bring server resources closer to the devices involved in the generation of data. It increases the intelligence of local area network by allowing computation of the data to be performed using the resource capabilities available inside the network where the data gathering devices exist. This helps to reduce latency in response times that is encountered in cloud computing where data was needed to be transmitted to servers placed in different geographical locations before any processing could begin. Fog computing has also allowed increased security of data by allowing highly sensitive data to be processed at fog servers and only low-sensitive data to be forwarded to the cloud server. It also promotes better management of huge volumes of data by distributing the data among multiple nodes in the local network.

Fog computing can be used in conjunction with cloud computing and edge computing. Cloud computing consists of multiple high-resource servers placed inside a data center owned by a service provider. Any user needing resources will associate with the provider and pay for the amount of resources used, without the need for delving into the details of managing these resources. Fog computing allows the processing to be performed at the local network of the data gathering devices. On the other side, edge computing allows the processing to be performed at either the devices that hold the sensors or a gateway node placed in close physical proximity to these sensor devices.

A. Vangala (✉) · A. K. Das
Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India
e-mail: anusha.vangala@research.iiit.ac.in; ashok.das@iiit.ac.in

C. Bhatt et al. (eds.), *Security Issues in Fog Computing from 5G to 6G*,
Internet of Things, https://doi.org/10.1007/978-3-031-08254-2_3

Internet of Things (IoT) is a collection of highly diverse devices with the ability to read the physical parameters of their surroundings, process the data in a distributed manner, and perform collective actions based on the processed data, using the Internet and with minimal human intervention. Fog computing has major applications in the IoT world where a huge amount of data sensed by the IoT smart sensor devices are regularly sent to the cloud servers for processing. The working of IoT is highly reliant on real-time processing of sensor data as the user is in continuous interaction with the smart devices. In such a scenario, latency due to processing and network transmission may be highly deterrent to the smooth functioning of many IoT applications. Also, data in IoT applications are sensitive to the user and require protection from any unprecedented misuse. This shows that fog computing is supremely relevant in the context of IoT applications.

Blockchain is compatible with fog computing as it allows the devices (nodes) to be used as blockchain nodes and fog nodes to be used as miner nodes. The idle resources available with the nodes in a fog network can be used for blockchain maintenance. The nodes that allow their idle resources to use for blockchain processing can be rewarded in accordance with the amount of resources provided, by using an appropriate consensus mechanism. In recent years, the blockchain technology has been adapted in many other potential applications in order to enhance the security of a system, such as smart farming [70, 71], IoT and industrial IoT [7, 58], Internet of Everything (IoE) [12], Internet of Drones (IoD) [11, 13, 14], smart grids [15], heathcare applications [36, 59, 63, 74], Internet of Vehicles (IoV) [6], Intelligent Transportation Systems (ITS) [69], Internet of Intelligent Things (IoIT) [80], Software-Defined Networks (SDN) [23], supply chains [43], and military applications [82].

3.1.1 Application Areas of Fog Computing

Nikouei et al. [52] proposed an authentication scheme on a smart surveillance system that is based on generating pattern indexes of identified interesting objects and timestamps on a live streaming video. The resulting indexes can be stored on the cloud to be used for further heavy processing. This event-oriented processing of the live video surveillance is done at three levels: a) object detection and tracking; b) extraction of low-level features at network edge; and c) data aggregation at fog nodes and processing and cloud centers. This requires event-oriented surveillance video query, real-time indexing, and secure data transferring, and blockchain-enabled authentication.

In the first level of object detection and tracking, the video live video is captured and sent to the edge or fog nodes in real time. The edge nodes then detect anomalies by extracting low-level features in the objects and behavior with minimal false alarm rate considering the limited resources available. This may require running a person, object, vehicle (POV) algorithm that is resource-heavy and hence avoided on edge devices. More resource-efficient tracker algorithms with the pre-trained

convolutional neural network (CNN)–Deep Learning models [3] are used. At the second level, certain relevant descriptive metrics are defined for the objects identified at the first level. Based on the defined metrics, further processing such as contextualization, classification, and saving are performed.

In general, the metric definition is done at the edge node, and the metric processing is outsourced to fog nodes or cloud servers. In such a case of outsourcing, the metric data needs to be transferred from the edge nodes and fog nodes and requires two-level encryption with symmetric encryption, such as Advanced Encryption Standard (AES) [2] and RSA public key encryption [57], with shared key encrypted with the fog node's public key, to prevent network sniffing attacks. This is initiated by the edge node that sends a handshake request to the fog node that obtains its public key certificate in response. The edge node sends the encrypted shared key. The fog nodes decrypt the shared key using its private key and send the hashed shared key. Once the edge node verifies the hashed shared key, data exchange can commence. The shared key is discarded at the end of every data exchange session. The features extracted at the edge node will be encrypted and forwarded to the fog node. The fog nodes then place a spatio-temporal context to the received features for contextualization. These frame-wise data with the location, time, sequence, the number of objects, and gestures are stored as key–value pair in fog nodes with sufficient storage that allows fast retrieval. This level of indexing speeds up the process of querying the video and replaces the slow process of observing the full video to identify the moments of interest.

The fog layer then shares the indexing data with the cloud layer for higher level processing tasks. To ensure a more secure and decentralized sharing with support to scalability, a blockchain-enabled authentication service is used. Every entity in the network has an account identified by its public key, called the virtual identity (VID) that is used in the identity authentication and management in the cloud server. When a fog node sends registration request to the cloud server, a profile is created after verifying the fog node's identity credentials. The hashed index table data is managed by a smart contract, which is deployed in the blockchain network allowing all the nodes to transparently access the transactions on the chain. Once the registration information of the fog node is verified, its access request is evaluated according to the authorization policies. If the request is granted, a transaction is executed by the cloud to update the list of entities authorized in the smart contract. Once the transaction itself is approved, the fog node receives the address of smart contract and the recording function. To authenticate the video query data stored on the fog, a cloud operator checks the current state of the smart contract and obtains the hashed key–value index record. Thus, the sensor devices can detect events in a video, which are indexed based on extracted features and stored in a table that is hashed to prevent malicious modifications.

Fernandez-Carames and Fraga-Lamas [34] provided a detailed study of introducing IoT with fog computing using blockchain in the field of education to propel educational sector toward smart universities and campuses. They defined the essential characteristics needed for such smart education system, compared different architectures provided for such a system, studied the effect of introducing

blockchain, analyzed the existing applications in smart education, and then proposed new challenges that should be researched in smart education. Chaiyarak et al. [20] also proposed an architecture for smart management of education even during unprecedented disastrous situations, such as the Coronavirus Disease-2019 (COVID-19) pandemic. In current situation, COVID-19 becomes a very serious health concern to the human life throughout the world [22]. One prominent solution is the use of the Internet of Medical Things (IoMT) that allows to deploy several wearable Internet of Things (IoT)-enabled smart devices in a patient's body [31, 36, 39]. The deployed smart devices should then securely communicate to nearby mobile device installed in a smart home, which then securely communicate with the associated Fog server for information processing. The processed information in terms of transactions is formed as blocks and put into a private blockchain consisting of cloud servers. Since the patient's vital signs are very confidential and private, the private blockchain is best suited for such kind of applications.

A number of smart IoT applications are engulfed in the concept of a smart city, such as smart lighting, smart transportation, smart healthcare, and smart buildings. Singh et al. [61] proposed an overview of such a smart city model and derived a blockchain and fog-based architecture with detailed characteristics of requirements along with a model diagram. It studies the average power consumption, based on the number of smart devices, and provides a latency comparison of fog and cloud systems in a smart city environment.

Islam et al. [42] proposed an architectural framework based on human activity recognition (HAR) directed toward monitoring patients with mental illnesses remotely. The activities of the patient captured through video are analyzed based on multi-class categorization using support vector machines. The accuracy of this classification is improved with the addition of blockchain-based fog architecture.

Gul et al. [38] proposed a reward-based business model based on blockchain that predicts medical status about a patient. Fernandez-Carames and Fraga-Lamas [35] proposed a communication architecture to remotely monitor the glucose levels of patients continuously and warn the patient to take appropriate preventive measures. This architecture makes use of crowdsourcing in mobile health for distributed problem solving, and federated blockchains are used to decentralize the system against single point of failure and increase the transaction privacy as the transaction data include highly sensitive medical data about patients. Fog computing is used in order to collect sample data from the patients using distributed mobile smart phone systems.

Baniata et al. [9] proposed a task scheduling system that can be used to efficiently automate the scheduling of tasks in complex applications such as smart city where task scheduling is considered as NP-hard problem, which is a computationally infeasible task. This system uses an ant colony optimization (ACO) on fog computing assisted with blockchain technology that is highly privacy-aware and takes very less execution time along with tackling high network load.

3.1.2 Main Contributions

In this chapter, we provide the following main contributions:

- We first discuss the necessity of security in fog computing environment. It is needed mainly due to the fact that the data is required to be protected from several potential attacks against passive as well as active adversaries.
- We then discuss the evolution of blockchain in fog computing context.
- Various security and functionality requirements in fog computing environment are discussed.
- Next, we discuss a taxonomy of various security protocols in fog computing. Design of security protocols for communication in fog computing may fall into one or more security protocols.
- We also discuss the network and threat models that are useful in discussing the existing security protocols for blockchain-enabled fog computing environment.
- Finally, a comparative study among the discussed existing security protocols for blockchain-enabled fog computing environment has been conducted to measure effectiveness of the protocols.

3.1.3 Chapter Organization

The remainder of this chapter is organized as follows. The security vulnerabilities of fog computing are discussed in Sect. 3.2. Section 3.3 studies in detail how blockhain can be integrated with fog computing and the evolution of this process since the inception of the ideas of blockchain and fog computing. The security requirements essential in fog computing are analyzed in Sect. 3.4. Section 3.5 is dedicated to the study of the types of security protocols needed to be designed keeping in view the security vulnerabilities and requirements of fog computing. Section 3.6 enhances the generalized architecture for fog computing by incorporating blockchain technology. It also studies the different threat models that apply to such a blockchain-based architecture that can help us analyze the existing security schemes. Section 3.7 studies a plethora of security schemes in detail that have been developed for fog computing using blockchain technology. Section 3.8 examines the security strength of studied schemes. Section 3.9 concludes the chapter by summarizing the blockchain-based fog computing solutions.

3.2 Need for Security in Fog Computing

Delegation of tasks to fog servers may cause some of the data to be available to these servers. Such data need to be protected against many attacks as defined in Sect. 3.4. Since fog servers are closer to the end users in the terminal layer, the surveillance

of the devices is relatively weak. This presents the requirement for better protection as the fog devices are much more prone to malicious attacks.

Stojmenovic et al. [65] presented a case study of how man-in-the-middle (MiTM) attack affects the system security in fog servers as the authors believe that this attack can potentially become the most common attack in fog computing. An experimental study is conducted by launching MiTM attack in four chronological steps to hijack communication in a fog-based system. They also studied the effect of intrusion detection based on anomaly detection by observing the memory consumption and CPU utilization of gateway node during the launched MiTM attack.

Ali et al. [4] studied that trust can be achieved only after the security goals of authentication, authorization, and privacy are achieved for each component in each level of the fog environment. Once trust is achieved, some dynamic method of identification is applied to each of the components in the environment. Butun et al. [18] mentioned that IoT as an environment is naturally prone to violation of user privacy due to the deep commingling of the user devices with other devices in the network. When such an environment is coalesced with a fog environment, the privacy violation is exacerbated due to the escalated complexity of determining the ownership of the huge amount of data circulated in the network.

Kaur et al. [44] identified that most of the security issues in fog computing correspond to the handing over of pre-processed data from the fog layer to the cloud layer address the need for lightweight security schemes compared to heavyweight schemes due to the significant resource limitations in the fog servers in the fog layer compared to cloud servers in the cloud layer. Mukherjee et al. [51] studied the comparison of cloud, edge, and fog computing along with the fog–cloud interface that allows the cloud layer to distribute the services to the fog servers in the fog layer in a resource-efficient manner.

From the above studies, it is clear that the security plays a very important role for protecting data in fog computing setting. The security of fog computing can be further enhanced by using the blockchain technology.

3.3 Blockchain and Its Evolution in Fog Computing

Baniata et al. [8] provided a detailed study on the integration of blockchains with fog computing and made observations that a majority of the applications of blockchains in fog computing were targeted toward maintaining data, followed by identity management, payment/trading, and reputation systems in IoT-related systems and used proof-based consensus algorithms with most of the applications using the Proof-of-Work (PoW) consensus protocol [10]. Integrating blockchain with fog computing requires a trade-off decision between the need for security, reliability, and decentralization with the cost of money, energy, and latency of using blockchains.

Uriarte et al. [68] studied the three blockchain-based fog solutions, a decentralized supercomputer, named Golem Network, a decentralized cloud named iExec

and a decentralized fog computer, named SONM, and identified that none of these solutions provide smart contract-based quality of service (QoS) and that privacy of consumer data is at stake since it is managed by their parties. The need for decentralized races was also identified.

Bouachir et al. [17] studied the challenges presented in a cyber-physical system useful for IoT and industrial Internet of Things (IIoT) and the usage of blockchains to overcome these challenges. They identify that the limited computational, communication, and storage resources of small devices are not naturally compatible with the blockchain infrastructure, which usually require high-compute-intensive machines. Also, blockchains are designed to use homogeneous nodes with equal capabilities and responsibilities, whereas the cyber-physical system environment has heterogeneous devices interconnected. Thus, the centralized network architecture is shifted to a distributed architecture with fog computing to overcome the resource limitations and heterogeneity challenges.

Wu et al. [83] proposed a strategy to integrate blockchains with fog computing by partitioning the fog server nodes into clusters such that every cluster has an associated access control list (ACL) stored on a customized compute-efficient blockchain to monitor and restrict access to resources between clusters. Figure 3.1 shows the evolution of blockchain in fog computing, which shows how the blockchain

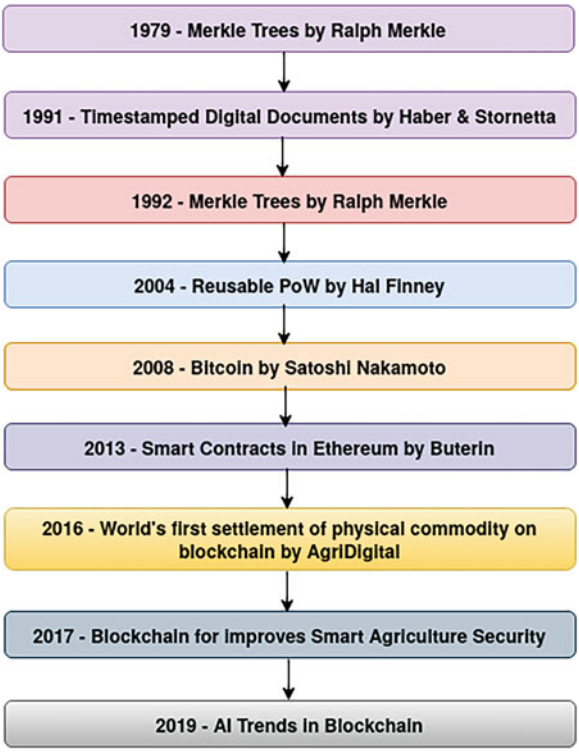

Fig. 3.1 Evolution timeline of blockchain in fog computing

technology started in 1979 and Artificial Intelligence (AI) trends came recently in blockchain for Big Data analytics purpose for accurate and better predictions on the data that are stored into the blockchains.

3.4 Security and Functionality Requirements in Fog Computing

The security requirements in fog computing environment are given below [64]:

- *Confidentiality:* It must be ensured that the data flowing in the network is understood by intended recipients only.
- *Data integrity:* Any message from an authorized sender to an intended recipient must not be altered during the transit.
- *Authentication:* It is required to validate a communicating node who it claims to be. All parties involved in fog computing environment, such as user, IoT smart device, fog node, and cloud server, must establish bi-directional trust through mutual authentication before granting access to restricted resources or any sensitive information.
- *Authorization:* In fog computing, where access control mechanisms are employed, it is required to authorize an authenticated entity to check if he/she has required privileges to access the requested resource. This, unauthorized access leads to an under-privileged user accessing an elevated resource.
- *Availability:* This requirement ensures that the services of fog computing are always available and must not be hampered by internal/external attacks or by resource starvation due to complex operations. In other words, denial-of-service (DoS) attacks must be prevented.
- *Data freshness:* Freshness is a very serious security feature in fog computing to ensure that the received data is freshly generated by the authentic participant and is not a replay message by an adversary.
- *Anonymity and privacy*: Identities of the entities must not be exposed to any eavesdropping adversaries.
- *Non-repudiability*: Every session must be uniquely associated with a valid communicating entity such that in case of misuse, the guilty can be held responsible for his/her actions.

Apart from the above security requirements, the following security properties should be fulfilled:

- *Forward secrecy:* If a node or an entity leaves the network, it must be blocked from reading any communication flowing in the network after its departure.
- *Backward secrecy:* If a new node (entity) joins the network, it must also be blocked from reading/decrypting the communication that is flowed before its introduction.

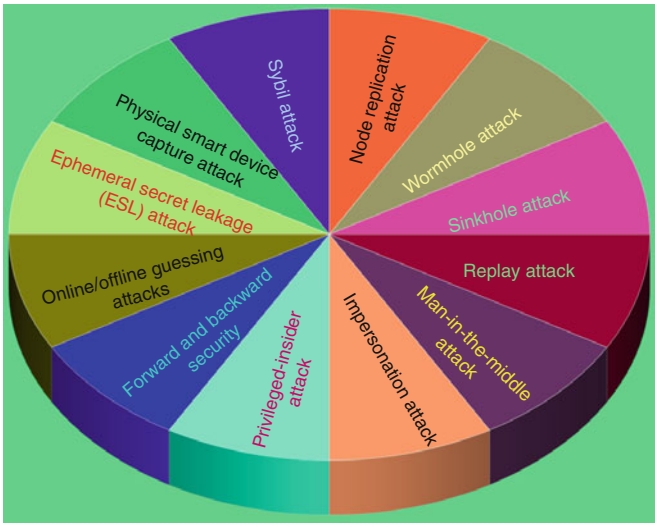

Fig. 3.2 Various possible potential attacks in fog computing environment

Additionally, the following attacks must be prevented in fog computing environment (see Fig. 3.2):

- *Node replication attack*: The attacker can deploy a malicious node that can simulate the identity and working of an existing node. The malicious node may generate fake messages in the network causing the other nodes to receive multiple conflicting messages.
- *Wormhole attack*: The adversary directs the messages between two nodes in the network such that these messages are tunneled through a set of nodes that are under the attacker's control. It forces the end nodes to misinterpret the route as the more efficient route by deceiving the end nodes into construing their distance between them as minimal. This allows for the network traffic to be shaped according to the attacker's needs. Such an attack can make provision for other attacks on the network traffic such as sniffing, modification, and dropping.
- *Sinkhole attack*: In this attack, an attacker compromises a node and modifies all routes to be directed through it so that all the traffic can be captured. This is done by publishing a less hop distance to misguide the neighbor nodes. Once the malicious node receives the traffic, it can misuse the re-directed traffic to eavesdrop, capture, modify, delete, or add messages to the traffic.
- *Replay attack*: The attacker monitors traffic between two communicating entities and copies certain message packets from sender to the receiver. These copied packets can then be sent to the receiver node multiple times to obtain undue advantage in terms of financial gain.
- *Man-in-the-middle attack*: This is a very specific attack in which the adversary first captures and blocks messages from the message sender to the message

receiver. Then the attacker creates counterfeit messages to be sent to the receiver. Similar action is repeated during the response from the receiver to the sender. This results on the counterfeit messages to be exchanged between sender and receiver instead of the real messages and allows the attacker to manipulate the two parties into thinking that they have exchanged the data with each other when in reality they have exchanged the data with the adversary. The two parties may not even be aware of the existence of the adversary.

- *Impersonation attack*: In this attack, the adversary illegitimately obtains the credentials of a legitimate entity. These credentials are then used by the attacker to communicate with other entities, misleading them into thinking that they communicate with the real entity.
- *Privileged-insider attack*: This attack is different from other attacks in that the adversary is not an outsider, but a legitimate user who has misuses his/her access privileges to obtain illegal information.
- *Online/offline guessing attacks*: Offline guessing attack refers to the act of speculating the correct login credentials of an entity. Online guessing attack is similar except that the adversary also attempts to login to the server. Offline guessing attack is considered to be more dangerous as the only limitation is the speed of the computer that is used to crack the password, whereas the speed of the network is an additional limiting factor in an online password guessing attack.
- *Ephemeral secret leakage (ESL) attack*: Any secret that is produced during the key establishment phase is called an ephemeral secret. Such secrets lead usually to play an important part in formulating the secret key and hence can present a major vulnerability in leakage of secret key information. ESL attacks are directed toward extraction of such ephemeral secrets used in the key agreement/establishment process.
- *Physical smart device capture attack*: This attack is possible in small-sized devices that may be mobile or immobile. The adversary seizes a device and extracts information from its memory. Such devices usually store secret information in their memory. If the device memory is insecure, this attack may lead to loss of a lot of secret information. Recovery from this attack involves replacement of such a device that may affect the cost involved.
- *Sybil attack*: In this attack, a malicious node maintains multiple active pseudonymous identities to itself in the network. Other nodes identify each of the identities to be unique nodes.

The following are the functionality requirements that are needed in fog computing deployment:

- A designed security protocol must be efficient in terms of storage, computation, and communication.
- Various entities registration/enrollment process should be executed in offline mode by the registration authority in order to reduce huge communication and computational overheads as the registration process is typically one-time procedure.

- The designed security protocol must support dynamic addition of entities in fog computing environment because some resource-constrained devices, such as IoT smart devices deployed in the network, may be physically captured by an adversary or they may be drained out of their battery power.
- A legal registered user must be permitted to change his/her password locally without contacting the registration authority in the designed security protocol.
- The designed security protocol must be scalable for supporting a huge number of nodes in a target network.

3.5 Taxonomy of Security Protocols in Fog Computing

Design of security protocols using the blockchain technology for communication in fog computing may fall into one or more of the categories as shown in Fig. 3.3.

3.5.1 Authentication

Authentication of an entity verifies the identity of that entity by comparing the given credentials associated with the entity with the existing credentials that are allowed to access the system. The entity under consideration may be a device, a host, or a user. Authentication of a message ensures that the origin of the message is the intended source entity. On the other side, authentication of an entity may be single factor or multi-factor. Single-factor authentication uses one set of credentials, whereas multi-factor authentication uses multiple sets of credentials to verify an identity. In general, up to three-factor authentication is commonly used. The classification of authentication is provided below:

- *User authentication*: Under this category, a user is typically registered with a trusted registration authority (RA) and obtains the secret credentials from the

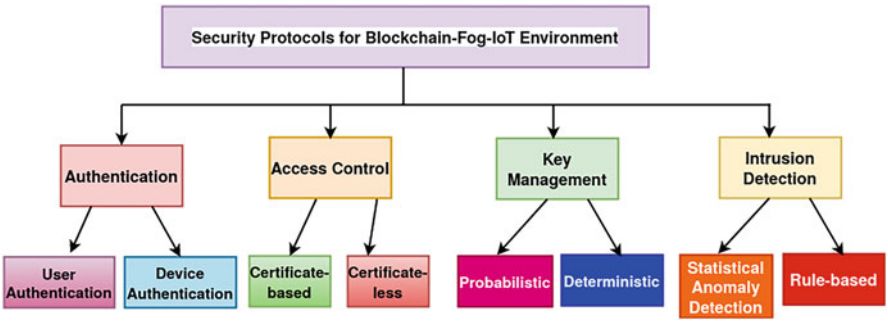

Fig. 3.3 Taxonomy of security protocols in fog computing environment

RA that are stored in a smart card or a mobile device. Later using the registration credentials stored in the smart card or mobile device, a user authentication with an accessed entity in fog computing environment and after successful mutual authentication, they establish a session key that is further used in secure communications [21, 30, 50, 66, 73, 77, 78].

- *Device authentication*: In device authentication, after registration with the RA, two devices need to mutually authenticate each other prior to establishment of session (secret) key using their pre-loaded registration credentials [81]. Next, using the establishment secret key, they can securely communicate each other for accessing the services in fog computing environment.

3.5.2 Access Control

Access control is the process of defining the operations that are allowed by an authorized entity in a given system and verifying that the entity is performing only the allowed operations on the system. Such access control schemes may use a certificate or they be also certificate-less. The access control mechanism primarily comprises the following two tasks [24, 25, 40, 45]:

- *Node authentication:* The newly deployed node must authenticate itself to the neighbor nodes in order to prove that it is a legal registered node and can access the network.
- *Key establishment:* It is essential for the newly deployed node in order to establish secret keys with the neighbor nodes to assure secure communication while transmitting the data only after mutual authentication.

3.5.3 Key Management

Two or more entities that wish to communicate securely with each other in such a way that the exchanged data is not visible to another external party must encrypt the data with a secret key common to all the entities involved in the communication. Such a key is to be agreed upon by all the involved entities and distributed securely among them [26–29]. This key also needs to be protected against compromise from different attacks. If it is compromised, the copy of the key at every entity must be replaced with a new agreed key. There are two types of key management that are possible:

- *Probabilistic key management*: Let the probability of establishing a secret key shared between any two neighbor nodes in the network be denoted by p_{key}. If $0 < p_{key} < 1$, a key management scheme is said to be probabilistic or randomized key management scheme.

- *Deterministic key management*: If $p_{key} = 1$, a key management scheme is termed as a deterministic key management scheme.

A node in fog computing environment (for example, an IoT smart device) can be physically captured by an adversary. By compromising the secret credentials stored in the compromised nodes, the attacker may be able to decrypt secure communication among other two non-compromised nodes in the network. Let $P_e(n_c)$ denote the fraction of secure communication links that are compromised when n_c nodes are already compromised in the network excluding the communication links that are directly involved due to compromise of n_c nodes. If $p_e(n_c) = 0$, we say a key management scheme is *unconditional secure* or *perfectly resilience* against physical node capture attack.

3.5.4 Intrusion Detection

Intrusion detection system (IDS) is a regular monitoring system that can be either a hardware device or a software to identify any activity that can be considered as malicious according to pre-defined rules or policies. The techniques to detect intrusion in a system can be statistical- or anomaly-based and rule- or signature-based [55, 56, 75, 76, 79]. In statistical techniques, behavior of the system under normal circumstance is defined and stored in the IDS. This is done by collecting relevant data of regular users who are allowed by the system as legitimate. While the system is monitored, its behavior is analyzed against the stored conditions to categorize the current condition as normal or abnormal, if it falls outside the scope for the defined behavior for normal working of the system. In the rule-based techniques, the behavior of the system under potential attack is defined. While the system or network is under surveillance, any activity that concords with the attack pattern is categorized to be an intrusion.

3.6 System Models

In this section, we elaborate the network and threat models related to blockchain-enabled fog computing environment.

3.6.1 Network Model

The network model exhibited in Fig. 3.4 consists of three layers: (1) terminal layer, (2) fog layer, and (3) cloud layer. These layers have the following functionalities and characteristics:

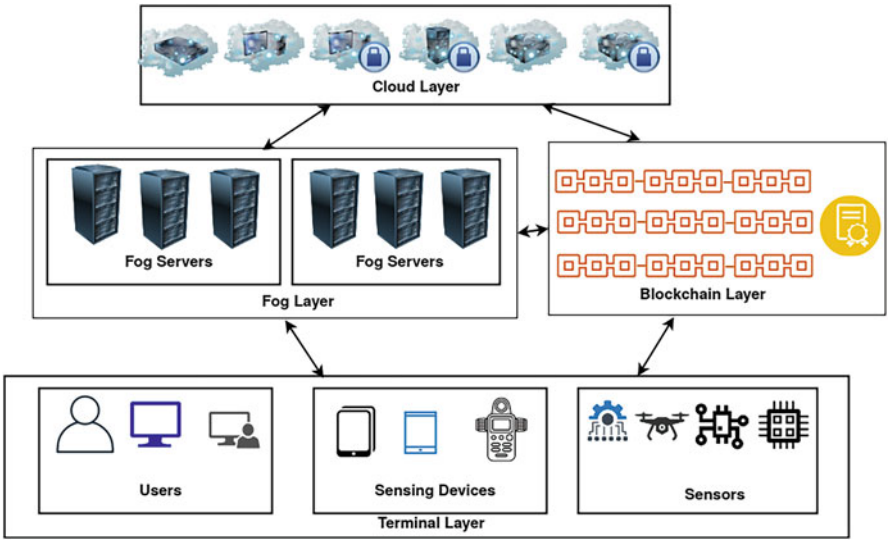

Fig. 3.4 Network model for a blockchain-enabled fog computing environment

- The *terminal layer* is the layer that consists of the end users, sensors, and other sensing and actuator devices that have the capability to sense the environmental readings of various physical parameters and send these parameters to the nearest fog server in the fog layer. It may also consist of actuator devices that can receive signals to perform a certain action that can affect the physical parameters.
- The *fog layer* consists of fog servers that are placed in groups nearer to the location of the actual sensor devices environment. These fog servers receive the sensor data consisting of readings of physical parameters and can perform certain pre-processing operations on this data. The blockchain is accessible to the terminal layer, fog layer, and also the cloud layer. Meta-data about this pre-processed data may be stored on the blockchain with the help of pre-defined operations in smart contracts. Once pre-processing is completed, the sensor data is stored at the fog servers until sufficient data is collected to be passed onto the cloud layer.
- The *cloud layer* consists of an assemblage of different types of cloud centers, such as private cloud center, public cloud center, and hybrid cloud center, which consist of high-end servers with access to private, public, and hybrid blockchains in the blockchain center. Depending on the application at hand, either private or public or hybrid blockchain may be used. The servers in the cloud layer have the capability to completely process the sensor data. Based on this processing, the smart contract may trigger a particular action to be performed at the terminal layer so the physical parameters can be controlled as required. This data may also be further forwarded to an *Artificial Intelligence (AI) and Big data center* that has the capability to utilize the data for further analysis and apply prediction

techniques to forecast any consequences of current actions at the terminal layer. Such consequences may trigger a new cycle of terminal to fog layer and fog to cloud layer data exchange and processing. Thus, the network model presents a system that is continuously active for the purpose of keeping the environmental surroundings in required limits.

3.6.2 Threat Model

The proposed network model is designed to be resistant against attacks defined in the threat models of Dolev–Yao model (DY model) [33] and Canetti–Krawczyk adversary model (CK model) [19].

- The DY model considers an adversary with the capability to monitor the messages communicated between two parties in order to modify or delete the messages. It also allows the adversary to add malicious content into the message on transit that may lead to misinterpretation of messages between the two parties.
- An adversary in the CK adversary model is similar to an adversary in the DY model. Apart from that, it can extract the information regarding the secret credentials, secret keys, and also any information about the states of the current session if all this information is stored insecurely in the memory of the communicating entities by launching session hijacking attack.

In addition, the end-point entities involved in communication are not in general trustworthy. The adversary may launch power analysis attack [49] or timing attack [32] to extract sensitive information from the physical captured devices' insecure memory and use the extracted data to impersonate the compromised entity. The cloud and fog servers can typically treated as semi-trusted entities in the network. Finally, the registration authority involved during the registration process is considered as a fully trusted entity in the network.

3.7 Security Solutions for Blockchain-Based Fog Computing Environment

The user authentication system proposed by Almadhoun et al. [5] uses smart contracts to map fog nodes with IoT devices [47]. The access control permissions of the users and their permitted operations are handled by the administrator. The end users access the devices with their unique Ethereum addresses indirectly via smart contract or directly though application. Their scheme supports confidentiality, integrity, and non-repudiation, and it is also resilient against denial-of-service (DoS) attacks [62]. This scheme is a traditional authentication scheme where the entity is to be declared legitimate prior to any exchange of communication data. Once

declared so, the entity is trusted to be authentic. This may lead to vulnerabilities toward attacks aiming on active sessions.

Al-Naji and Zagrouba [41] builds upon the scheme by Almadhoun et al. [5] by adding continuous authentication, in which the entities involved in a session are continuously authenticated for the duration of an active session. This scheme is developed as a user-to-device model for mutual authentication between an end user and fog nodes with a smart contract issuing the access to avoid the involvement of trusted third party. A machine learning model for face recognition is used at the fog layer that is continuously updated according to the data collected by the IoT smart devices. The smart contract applies face similarity score and a similarity threshold on the face recognition model to obtain the trust model based on the comparison results, which then yields the access decision model with the decision to continue or lockout that is fed back into the fog layer.

Wang et al. [72] proposed a mutual authentication scheme between an end user EU_i and an edge server ES_j with the key materials table $KMST$ deployed on a smart contract over a blockchain system based on Ethereum or Hyperledger fabric. The deployed smart contract contains algorithms to perform initialize, update, query, and revoke on the $KMST$. A trusted registration authority (RA) registers the end user EU_i and the edge server ES_j via separate private and secure communication channels between them. When an EU decides to join the network, it sends a request with ID_i to RA. The RA chooses its own private scalar $r_i \in Z_q^*$ and multiplies the base point P, r_i times, to obtain a point on a non-singular elliptic curve as $R_i = r_i.P$, where $k.P = P + P + \cdots + P(k\,times)$ denotes the elliptic curve point (scalar) multiplication [46], q is a large prime such that elliptic curve discrete logarithm problem (ECDLP) becomes intractable, $Z_q = \{0, 1, \cdots, q-1\}$, and $Z_q^* = \{1, \cdots, q-1\}$. RA computes another private scalar $x_i \in Z_q^*$ for EU_i using its own private scalar r_i and its own master key s, and also computes the corresponding public key for EU_i as PK_i. RA generates PID_i as the hash of EI_i's public key PK_i and encrypts EU_i's identity ID_i. x_i is stored securely at EU_i. Similar procedure is applied at ES_j to obtain, verify, and store x_j.

The authentication process in Wang et al.'s scheme [72] between EU_i and ES_j starts with EU_i generating a private scalar $a \in Z_q^*$ and computing the corresponding elliptic curve point $A = a.P$. It then computes parameter pid_i as the bitwise exclusive-OR (XOR) of PK_i and the hash of A concatenated with the point $a \cdot PK_j$. It also computes the parameter k as the sum of a and the product of x_i and the hash of PK_i, pid_i, A and the timestamp t_i. EU_i sends the parameters A, pid_i, k, and t_i to ES_j. ES_j verifies timestamp t_i, extracts PK_i from pid_i, and hashes it to obtain PID_i that is sent as an argument to query the KMST and check the validity. If the result is true, ET_i has not expired and ES_j verifies the parameter k using A and PK_i extracted above. ES_j then computes the parameters K_1 as the sum of the points obtained from point multiplication of x_j, A and b, PK_i, where b is a private scalar chosen by ES_j and its equivalent ECC point computed as B. The other parameter $K2$ that is the last component of the session key is obtained as the product of private scalar b and the point A received from EU_i. The point B,

the session key verifier w, and timestamp t_j are sent to the EU_i. EU_i computes the session key similar to ES_j and verifies it using the session key verifier.

Pallavi and Kumar [53] proposed an authentication scheme based on smart contracts that allows the data owners to verify the entities requesting for the data without the involvement of any third party. The proposed scheme uses a system model consisting of an administrator, end users, fog nodes, IoT smart devices, and cloud. Administrator registers fog nodes and IoT smart devices and handles access control through attribute permission using smart contracts. The end users are the requesters of the data from specific IoT smart devices. Fog nodes provide some storage and computation ability to reduce the processing and storage latency at the IoT devices. The cloud consists of the complete collection of data that can be used for heavy processing. A smart contract consists of a mapping of which IoT smart devices send their data to specific fog nodes and also a mapping of which end users are allowed to access which IoT smart devices. To register a device, the admin creates a smart contract that generates a device password, based on the device ID, and Ethereum address also uploaded to the IoT smart device. The device password is also recomputed and stored at the smart device. To access the device, an end user needs to specify this device password correctly. To map the fog nodes and IoT smart devices, admin creates a message using device Ethereum address and another message using Ethereum address of fog node. The pair of fog node and the device to be mapped are then stored in both fog nodes and IoT smart devices. Similarly, another message is created from user ID and password that is stored in both end user and smart device to map them together. The authentication phase authenticates the end user to the admin by sending a request to access a specific device with its Ethereum address. The request may be rejected by the admin if the user is not authorized to access the requested device. The token generation phase uses the hash of timestamp, device Ethereum address, and public key to create: (a) an access user token with the device Ethereum address and identity and (b) a user token with the user identity and Ethereum address. As a response to the authentication request sent in the authentication phase, the admin sends an access token to the user followed by the creation of user token by the user. The tokens are verified by the smart contract by computing a message that encrypts the product of user password XORed with the timestamp and the hash of device identity concatenated with the random number used in the first message of device–fog mapping. After the end user enters the user password for access, the verification message is also computed at the end user. If the verification messages generated at the smart contract and the end user are identical, then the user token is sent to the fog node and an access token is sent to the end user. The received tokens are verified at both ends. From the received user token, the fog node computes the user private key and the session key as the encryption of the device identity concatenated with device password. The fog nodes send both the private key and the session key to the end user. The end user digitally signs its user token with the received private key and sends to the fog node. The fog node then verifies the digital signature on the user token and generates the first signed message with encryption of user token and user private key concatenated, and a second signed message as the hash of the Chebyshev polynomial [37] XORed with

the user token. The end user sends the two signed messages to the fog node. The fog node re-computes the two signed messages and verifies if the received signed messages match the computed messages. If so, the end user is granted access to the IoT smart device. In the data exchange phase, the end user and IoT device can directly communicate over an established secure sockets layer (SSL) connection. The fog node and the end user generate a parameter by encrypting the concatenation of the first signed message and the session key and adding it to the user private key. If this computed parameter matches both the fog node and the end user, the data exchange can be successfully initiated. The drawback of this scheme is that the fog nodes send both the private key and the session key to the end user. This scheme requires the channel to be a private secure channel (via SSL) between the fog nodes and the device.

Abdalah et al. [1] proposed a system where a controller registers and manages the gateway fog nodes and registers the IoT smart devices. Each controller manages one gateway fog node, and each fog node manages multiple IoT smart devices. The cloud server is a centralized system that has the capability to register the devices, create users, deploy smart contract, and register the controller to the blockchain. The cloud server first registers the devices by executing the add device function in the smart contract and issues a private key to the device. The device itself generates the corresponding public key from the private key received. The device now registers with the gateway fog node by sending a request with its public key and receives the gateway public key in response. To prevent replay attack, the device sends its identity, its device information in JavaScript Object Notation for Linked Data (JSON-LD) format [67], a nonce, and a timestamp encrypted with the gateway's public key to the gateway node. After verifying the timestamp, the gateway forwards this request to the controller that ensures that the device exists of the blockchain and is registered by the user before adding the device identity and the device information onto the blockchain. Once it is done, the controller sends the device identity and a new nonce encrypted with the gateway public key to the gateway. If the nonce is valid, the gateway replies to the device with the device identity and user–device key. The device is now registered. The gateway sends its identity and public key to the controller, which invokes the smart contract to add the gateway and responds with its own public key if the gateway registration is successful. During device authentication process, the device sends its identity and request nonce to the gateway, which invokes the get device function in smart contract to extract the device information, if it is legitimate. The gateway responds to the device with the received request nonce and adds a new response nonce encrypted with the user–device key. The device decrypts the response nonce with the user–device key and sends this nonce to the gateway encrypted with the gateway's public key. If this is verified correctly, the gateway responds with the request nonce and a timestamp to declare that the device is now authenticated to communicate with the device.

Patonico et al. [54] proposed an authentication scheme designed to provide anonymity and data integrity along with the generation of a secret session key by computing separate parameters for each of these security functionalities. Each device is pre-loaded with an identity, a certificate, a pair of public and private keys,

and the public key of the cloud server. The sum of a random variable and the private key of the computing entity is generated. This sum is multiplied with the base point of an elliptic curve to obtain the parameters for session key generation. The sum is further multiplied with the public key of the cloud server to obtain a symmetric key. The first anonymity parameter is generated by encrypting the entity identity and certificate using the symmetric key. A second anonymity parameter is generated by multiplying the sum with public key of the entity. The data integrity parameter is generated by hashing the concatenation of session key parameter, anonymity parameters, and data integrity parameter. All these parameters are sent to the fog device, and the sensor device is said to be initialized at this stage. The fog server follows a similar procedure to obtain its own session key parameters, anonymity parameters, and data integrity parameters with the first anonymity parameters generated using session key parameters and second anonymity parameters of both sensor node and fog node. All the parameters from fog node, the second sensor anonymity parameter, and the sensor session key parameter are all forwarded to the central server. The central server follows the same procedure as the fog node and generates an anonymity parameter for the sensor device that is passed to the sensor node via the fog node. The session key consists of the hash of the session key parameter between sensor-to-fog, fog-to-central server, and central server-to-sensor node.

Yang et al. [84] proposed a framework for access control with a cloud service provider, a data owner that uploads the data to the cloud, and the associated access rights for each resource to the blockchain and a data user that accesses resources from the cloud if verified for the access rights requested. The need for blockchain arises from the fact that the cloud is assumed to be only semi-trusted. When the data user requests for a resource from the cloud, it queries the blockchain for the access rights of the user for the requested resource. Depending on the result obtained from the blockchain, the final access permission is determined. For the smooth flow in this system, the cloud, the data owner, and the data user register with the blockchain in the initialization phase by sending a request message along with the start and end timestamps of the time period during which the data requested is to be synchronized. The blockchain generates the public and private keys for the cloud using a smart contract function. Using the public key of the cloud, the associated address for the cloud is determined. The symmetric key between the cloud and the blockchain is used to encrypt the cloud keys pair, and the cloud address with the private cloud key encrypted again with the symmetric key. The cloud uses the symmetric key to decrypt its address and key pair. The data user and data owner are also registered in a similar procedure. To publish the resource, it is uploaded to the cloud by the data owner, and the associated metadata returned by the cloud is uploaded to the blockchain using a smart contract function. To access a resource, the user sends the encrypted resource information and its own address to the cloud. The cloud decrypts the resource information and the user address and obtains the hashed resource information that is passed to a smart contract function on the blockchain that returns the appropriate result metadata information, which is passed on from the blockchain to the cloud. The cloud decrypts this result metadata information

and checks if the resulting metadata information in the actual data in the cloud has the same value of this result metadata returned from the blockchain. If it happens so, the cloud responds to the user to access the resource and updates the access log in the blockchain about this recent user access to the resource. Authorization of access to different users may be directly given to the data owner by allowing the blockchain to call the verification smart contract function or indirectly by a previous data user to other data users by allowing the data users to send an authorization notice proving that they are allowed to authorize other data users. This scheme uses the address of resources and users instead of usernames that gives improved performance. In addition, this scheme provides accountability, availability, authenticity, and integrity with multiple protection mechanisms.

Zhang [85] proposed a key management scheme, named as dynamic contributory broadcast encryption, that can be used to establish a secure channel among a group of fog nodes such that a common public key is generated for encryption and separate private keys for each fog node in the groups are generated for decryption, without the involvement of any third party. This allows any external end user to generate messages intended to be received by one of the fog nodes in the group and securely encrypt it with the group fog public key. Such a message can only be decrypted by the specific recipient fog node in the group. No other node in the group will be able to decrypt the correct message. This scheme allows any fog node to leave the group at any time and any node to join the group at any time. This scheme uses the bilinear pairing cryptographic primitive to generate a tuple for each group of fog nodes corresponding to the group size. A bilinear pairing is a mapping e: $G_1 \times G_1 \rightarrow G_2$ with the following three properties [16, 48]. Here, G_1 and G_2 are the cyclic additive and multiplicative groups of a large prime order q, and G_2 is called the target group.

- **Bilinearity**: $e(P + Q, R) = e(P, R)e(Q, R)$ and $e(P, Q + R) = e(P, Q)e(P, R), \forall P, Q, R \in G_1$. In general, we have $e(aP, bQ) = e(P, Q)^{ab}$, $\forall a, b \in Z_q^* = \{1, 2, \cdots, q - 1\}$.
- **Non-degeneracy**: Let e_{G_1} be the identity in G_1. Then, $e(P, P) \neq e_{G_1}, \forall P \in G_1$.
- **Computability**: There is an efficient polynomial-time algorithm to calculate $e(P, Q), \forall P, Q \in G_1$.

The group size dynamically determines based on the earlier groups and applications. Since the fog nodes can join and leave dynamically, the number of fog nodes may exceed the group size at some point that necessitates the need for creation of a new group. Every fog node is given a position in the system. A system position is set to 1 if occupied by a fog node. During initialization, every fog node computes its local parameter for itself and other fog nodes and publishes all these parameters. The rest of the fog nodes then computes the public encryption key from the received messages and derives its own private decryption key. When a new fog node is to join the system, it has to repeat the initialization process and set its position in the system to 1. When a fog node is to leave the group, it publishes its local parameters for itself and its group nodes to the entire group. The rest of the members then multiplies with the inverse of these parameters to nullify the existence of this fog node. To send a

message to multiple groups, an end user has to encrypt the same message using the public keys of the intended groups multiple times separately. This process has a large communication overhead, and to reduce this, a uniform session key for all the groups can be shared to every group encrypted with the group session key so that a broadcast message can be passed to all the fog nodes across all the groups at once. This trade-off reduces the complexity of communication to linear complexity.

Shabisha et al. [60] proposed an authentication system and key agreement system for a group of fog nodes where a fully trusted server registers and authorizes the devices and the fog nodes jointly perform the agreement of the key among themselves. The designed scheme, based on elliptic curve cryptography and Lagrange interpolation, considered several factors such as ensuing the privacy of the devices and keeping the connection among the nodes untrackable, with no necessity of pre-shared key variables. The scheme has been designed by considering two typologies: (a) static topology, where the devices have fixed locations and are statically assigned to a fixed fog node, and (b) dynamic topology, where the devices are assumed to be mobile leading dynamic mapping of fog nodes based on the changed location. The initialization phase ensures that the device's public session parameter is known to the fog node and the server with the device's identity hidden using hash and elliptic curve point multiplication with the server public key. The difference between the static and dynamic initialization is that the public device parameter is computed at the server for static initialization, whereas it is computed at the device node for dynamic initialization. The group authentication and key agreement phase runs in four stages: (a) request of update by fog node, (b) response by devices, (c) response by fog nodes, and (d) acknowledgment. In request of update stage, the fog node computes a signature on a public variable that is derived from a local private random variable and sends it to the server along with its fog identity. After the server verifies the signature, it computes its own signature from a local private random variable, two parameters for device and fog node, and a polynomial from the Lagrange interpolation. The hashed polynomial along with the server signature and fog signature are sent to the fog. The fog verifies the signature and forwards the message to the device. The device verifies the integrity of the message and the signature before storing the hashed polynomial. It encrypts the received parameters after hashing their concatenation and forwards them to the fog node. The fog node reconstructs the polynomial and extracts all the coefficients and forwards them to the device. Using this, the device derives the polynomial using the Lagrange interpolation and checks if it matches with the stored hashed polynomial. If it is so, the local private random variable is updated, and the hash of the polynomial, old private random value, and new private random value are taken and multiplied with the device private key and subtracted from the new private random value to obtain the difference as s. This differences s along with the old and new private random values are passed to the fog node. The fog node performs the same computation of difference, and after verification with the received difference, it updates the public random variable of device in its memory. The server performs the same verification and updation of device public random variable in its memory. For data exchange, the message may be sent in plaintext or encrypted with the symmetric key

between device and fog node. A timestamp is further generated. Three hashes of the message and timestamp, the symmetric key and timestamp, and the message, the symmetric key, and timestamp are generated, concatenated, and encrypted with the Lagrange polynomial as the key for non-encrypted communication. For encrypted communication, the ciphertext corresponding to the message is concatenated with the hash of the symmetric key and timestamp, along with the Lagrange polynomial and passed to the fog node. This scheme provides authentication of the entities, authenticity, integrity, anonymity, unlinkability, perfect forward secrecy, group forward secrecy, and backward confidentiality.

3.8 Comparative Analysis

In this section, we perform a detailed comparative study on the communication and computational costs and also security features among various state-of-the-art security protocols, such as the schemes designed by Almadhoun et al. [5], Al-Naji and Zagrouba [41], Wang et al. [72], Pallavi and Kumar [53], Abdalah et al. [1], Patonico et al. [54], Yang et al. [84], Zhang [85], and Shabisha et al. [60].

3.8.1 Comparative Analysis on Communication and Computational Costs

For comparative study on the communication and computational costs, we have computed the communication and computational costs for different schemes. Next, we have rearranged the schemes in descending order based on their communication and computational costs. If the computational/communication cost for a scheme is high/very high, we have marked it as **high**; if the computational/communication cost of a scheme is low, we have marked it as **low**; otherwise, if the computational/communication cost for a scheme is medium, we have then marked it as **medium**. Table 3.1 shows a comparative study on communication and computational costs for the existing schemes.

The studied security protocols have been compared on the basis of the number of operations required for expensive computations in the schemes and the amount of data to be transmitted as communication costs. The cost ranges for communication less than 3000 bits have been considered as low, and more than 4000 bits has been taken as high. With computation cost, the schemes that are based on bilinear pairings or involve many elliptic curve multiplication operations turn out to have very high computation costs.

Table 3.1 Comparative study on communication and computational costs

Scheme	Communication cost	Computational cost
Almadhoun et al. [5]	Low	Medium
Al-Naji and Zagrouba [41]	Low	Medium
Wang et al. [72]	Medium	Medium
Pallavi and Kumar [53]	Low	Low
Abdalah et al. [1]	High	High
Patonico et al. [54]	High	High
Yang et al. [84]	Medium	High
Zhang [85]	High	High
Shabisha et al. [60]	Low	High

Table 3.2 Comparative study on security features

Features	[5]	[41]	[72]	[53]	[1]	[54]	[84]	[85]	[60]
Confidentiality	✓	✓	✓	✓	✓	×	✓	✓	✓
Integrity	✓	✓	✓	✓	✓	✓	✓	✓	✓
Authenticity	✓	✓	✓	✓	✓	✓	✓	✓	✓
Non-repudiation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Anonymity	✓	✓	×	✓	✓	×	×	×	✓
Traceability or unlinkability	✓	✓	×	✓	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓	✓	✓	✓	✓	✓
Key agreement	✓	✓	✓	✓	✓	✓	✓	✓	✓
Forward secrecy	×	×	×	×	×	×	✓	✓	✓
Backward secrecy	×	×	×	×	×	×	✓	✓	✓
Replay attack	✓	✓	✓	✓	✓	✓	✓	✓	✓
Man-in-the-middle attack	✓	✓	✓	✓	✓	✓	✓	✓	✓
Privileged-insider attack	×	×	×	×	✓	✓	×	×	✓
ESL attack	×	✓	×	✓	✓	✓	×	✓	✓
Impersonation attack	×	✓	✓	✓	✓	✓	×	✓	✓
Physical node capture Attack	×	✓	✓	✓	✓	✓	×	✓	✓
DoS attack	✓	✓	×	×	✓	✓	×	✓	✓

Note: ✓: A scheme resists an attack or supports a feature; ×: a scheme does not resist an attack or it does not support a feature

3.8.2 Comparative Analysis on Security Features

The security features among the existing schemes are compared in Table 3.2. Several security features have been considered based on security requirements and threats in fog computing environment that are already discussed in Sect. 3.4. It is evident that the scheme [60] provides better security as compared to other existing schemes considered in Table 3.2.

3.9 Conclusion

In this chapter, we focused on studying fog computing in detail by analyzing its applications in various fields. The applied analysis allows to understand the need for security in fog computing. Once the security and functionality requirements of fog computing were identified, the evolution of the usage of blockchains to fulfill the security gaps in fog computing was studied. The literature was analyzed to understand the existing security schemes that apply blockchains in fog computing. Finally, we provided a detailed comparative analysis on the communication and computational costs and also security features among various state-of-the-art security protocols that are proposed in the line of blockchain-based fog computing environment.

References

1. Abdalah, A. N., Mohamed, A., & Hefny, H.A. (2020). Proposed authentication protocol for IoT using blockchain and fog nodes. *International Journal of Advanced Computer Science and Applications, 11*(4)
2. Advanced Encryption Standard (AES). (2001). FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. Accessed on February 2021.
3. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *International Conference on Engineering and Technology (ICET)* (pp. 1–6). https://doi.org/10.1109/ICEngTechnol.2017.8308186.
4. Ali, A., Ahmed, M., Imran, M., & Khattak, H.A. (2020). *Security and privacy issues in fog computing* (chap. 5, pp. 105–137). Hoboken: John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119551713.ch5.
5. Almadhoun, R., Kadadha, M., Alhemeiri, M., Alshehhi, M., & Salah, K. (2018). A user authentication scheme of IoT devices using blockchain-enabled fog nodes. In *15th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, Aqaba, Jordan (pp. 1–8).
6. Bagga, P., Sutrala, A. K., Das, A. K., & Vijayakumar, P. (2021). Blockchain-based batch authentication protocol for Internet of Vehicles. *Journal of Systems Architecture, 113*, 101877.
7. Banerjee, S., Bera, B., Das, A. K., Chattopadhyay, S., Khan, M.K., & Rodrigues, J. J. (2021). Private blockchain-envisioned multi-authority CP-ABE-based user access control scheme in IIoT. *Computer Communications, 169*, 99–113.
8. Baniata, H., & Kertesz, A. (2020). A survey on blockchain-fog integration approaches. *IEEE Access, 8*, 102657–102668. https://doi.org/10.1109/ACCESS.2020.2999213.
9. Baniata, H., Anaqreh, A., & Kertesz, A. (2021). PF-BTS: A privacy-aware fog-enhanced blockchain-assisted task scheduling. *Information Processing & Management, 58*(1), 102393. https://doi.org/10.1016/j.ipm.2020.102393.
10. Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. (2014). Proof of activity: Extending Bitcoin's proof of work via proof of stake. *SIGMETRICS Performance Evaluation Review, 42*(3), 34–37.
11. Bera, B., Chattaraj, D., & Das, A. K. (2020). Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment. *Computer Communications, 153*, 229–249.
12. Bera, B., Das, A. K., Obaidat, M., Vijayakumar, P., Hsiao, K. F., & Park, Y.: AI-enabled blockchain-based access control for malicious attacks detection and mitigation in IoE. *IEEE Consumer Electronics Magazine*, 1–1 (2020). https://doi.org/10.1109/MCE.2020.3040541.

13. Bera, B., Das, A. K., & Sutrala, A. K. (2021). Private blockchain-based access control mechanism for unauthorized UAV detection and mitigation in Internet of Drones environment. *Computer Communications, 166*, 91–109.
14. Bera, B., Saha, S., Das, A. K., Kumar, N., Lorenz, P., & Alazab, M. (2020). Blockchain-envisioned secure data delivery and collection scheme for 5G-based IoT-enabled internet of drones environment. *IEEE Transactions on Vehicular Technology, 69*(8), 9097–9111.
15. Bera, B., Saha, S., Das, A. K., & Vasilakos, A. V. (2020). Designing blockchain-based access control protocol in IoT-enabled smart-grid system. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.3030308.
16. Boneh, D. (2012). Pairing-based cryptography: Past, present, and future. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'12)*, Beijing, China (pp. 1–1)
17. Bouachir, O., Aloqaily, M., Tseng, L., & Boukerche, A. (2020). Blockchain and fog computing for cyberphysical systems: The case of smart industry. *Computer, 53*(9), 36–45. https://doi.org/10.1109/MC.2020.2996212.
18. Butun, I., Sari, A., & Öhsterberg, P. (2020). Hardware security of fog end-devices for the Internet of Things. *Sensors, 20*(20). https://doi.org/10.3390/s20205729.
19. Canetti, R., & Krawczyk, H. (2002). Universally composable notions of key exchange and secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, Amsterdam, The Netherlands (pp. 337–351).
20. Chaiyarak, S., Koednet, A., & Nilsook, P. (2020). Blockchain, IoT and fog computing for smart education management. *International Journal of Education and Information Technologies, 14*. https://doi.org/10.46300/9109.2020.14.7.
21. Challa, S., Das, A. K., Gope, P., Kumar, N., Wu, F., & Vasilakos, A. V. (2020). Design and analysis of authenticated key agreement scheme in cloud-assisted cyber-physical systems. *Future Generation Computer Systems, 108*, 1267–1286.
22. Chamola, V., Hassija, V., Gupta, V., & Guizani, M. (2020). A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact. *IEEE Access, 8*, 90225–90265.
23. Chattaraj, D., Saha, S., Bera, B., & Das, A. K. (2020). On the design of blockchain-based access control scheme for software defined networks. In *IEEE INFOCOM 2020 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada (pp. 237–242). https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162669.
24. Chatterjee, S., Das, A. K., & Sing., J. K. (2013). Analysis and formal security verification of access control schemes in wireless sensor networks: a critical survey. *Journal of Information Assurance and Security, 8*(1), 33–57.
25. Chatterjee, S., Das, A. K., & Sing., J. K. (2014). An enhanced access control scheme in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks, 21*(1–2), 121–149.
26. Das, A. K. (2008). An identity-based random key pre-distribution scheme for direct key establishment to prevent attacks in wireless sensor networks. *International Journal of Network Security, 6*(2), 134–144.
27. Das, A. K. (2008). An unconditionally secure location-aware key management scheme for static sensor networks. *Journal of Discrete Mathematical Sciences and Cryptography, 11*(3), 333–355.
28. Das, A. K. (2008). ECPKS: An improved location-aware key management scheme in static sensor networks. *International Journal of Network Security, 7*(3), 358–369.
29. Das, A. K. (2012). A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. *International Journal of Information Security, 11*(3), 189–211.
30. Das, A. K., Sutrala, A. K., Kumari, S., Odelu, V., Wazid, M., & Li, X. (2016). An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks. *Security and Communication Networks, 9*(13), 2070–2092.

31. Das, A. K., Wazid, M., Kumar, N., Khan, M. K., Choo, K. K. R., & Park, Y. (2018). Design of secure and lightweight authentication protocol for wearable devices environment. *IEEE Journal of Biomedical and Health Informatics, 22*(4), 1310–1322. https://doi.org/10.1109/JBHI.2017.2753464.
32. Dhem, J. F., Koeune, F., Leroux, P. A., Mestre, P., Quisquater, J. J., & Willems, J. L. (1998). A practical implementation of the timing attack. In *International Conference on Smart Card Research and Advanced Applications* (pp. 167–182). Berlin: Springer.
33. Dolev, D., & Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory, 29*(2), 198–208.
34. Fernández-Caramés, T. M., & Fraga-Lamas, P. (2019). Towards next generation teaching, learning, and context-aware applications for higher education: A review on blockchain, IoT, fog and edge computing enabled smart campuses and universities. *Applied Sciences, 9*(21), 4479.
35. Fernandez-Carames, T. M., & Fraga-Lamas, P. (2019). Design of a fog computing, blockchain and IoT-based continuous glucose monitoring system for crowdsourcing mHealth. In *Proceedings of 5th International Electronic Conference on Sensors and Applications* (vol. 4). https://doi.org/10.3390/ecsa-5-05757.
36. Garg, N., Wazid, M., Das, A. K., Singh, D. P., Rodrigues, J. J. P. C., & Park, Y. (2020). BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for internet of medical things deployment. *IEEE Access, 8*, 95956–95977.
37. Gil, A., Segura, J., & Temme, N. M. (2007). *Numerical methods for special functions*. Philadelphia, USA: Society for Industrial and Applied Mathematics (SIAM). https://epubs.siam.org/doi/abs/10.1137/1.9780898717822.
38. Gul, M. J., Subramanian, B., Paul, A., & Kim, J. (2021). Blockchain for public health care in smart society. *Microprocessors and Microsystems, 80*, 103524.
39. Guo, R., Yang, G., Shi, H., Zhang, Y., & Zheng, D. (2021). O-R-CP-ABE: An efficient and revocable attribute-based encryption scheme in the cloud-assisted IoMT system. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2021.3055541.
40. Huang, H. F. (2009). A novel access control protocol for secure sensor networks. *Computer Standards & Interfaces, 31*, 272–276.
41. Hussain Al-Naji, F., & Zagrouba, R. (2020). CAB-IoT: Continuous authentication architecture based on Blockchain for Internet of Things. *Journal of King Saud University – Computer and Information Sciences*. https://doi.org/0.1016/j.jksuci.2020.11.023.
42. Islam, N., Faheem, Y., Din, I. U., Talha, M., Guizani, M., & Khalil, M. (2019). A blockchain-based fog computing framework for activity recognition as an application to e-healthcare services. *Future Generation Computer Systems, 100*, 569–578.
43. Jangirala, S., Das, A. K., & Vasilakos, A. V. (2020). Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment. *IEEE Transactions on Industrial Informatics, 16*(11), 7081–7093.
44. Kaur, J., Agrawal, A., & Khan, R. A. (2020). Security issues in fog environment: A systematic literature review. *International Journal of Wireless Information Networks, 27*, 467–483.
45. Kim, H. S., & Lee, S. W. (2009). Enhanced novel access control protocol over wireless sensor networks. *IEEE Transactions on Consumer Electronics, 55*(2), 492–498.
46. Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation, 48*, 203–209.
47. Kumari, A., Tanwar, S., Tyagi, S., & Kumar, N. (2018). Fog computing for Healthcare 4.0 environment: Opportunities and challenges. *Computers & Electrical Engineering, 72*, 1–13. https://doi.org/10.1016/j.compeleceng.2018.08.015.
48. Menezes, A. (2013). An introduction to pairing-based cryptography. https://www.math.uwaterloo.ca/~ajmeneze/publications/pairings.pdf. Accessed on May 2020.
49. Messerges, T. S., Dabbish, E. A., & Sloan, R. H. (2002). Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers, 51*(5), 541–552.
50. Mishra, D., Das, A. K., & Mukhopadhyay, S. (2016). A secure and efficient ECC-based user anonymity-preserving session initiation authentication protocol using smart card. *Peer-to-Peer Networking and Applications, 9*(1), 171–192.

51. Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., & Kumar, V. (2017). Security and privacy in fog computing: challenges. *IEEE Access, 5*, 19293–19304. https://doi.org/10.1109/ACCESS.2017.2749422.
52. Nikouei, S. Y., Xu, R., Nagothu, D., Chen, Y., Aved, & A., Blasch, E. (2018). Real-time index authentication for event-oriented surveillance video query using blockchain. In *2018 IEEE International Smart Cities Conference (ISC2)* (pp. 1–8). https://doi.org/10.1109/ISC2.2018.8656668.
53. Pallavi, K. N., & Kumar, V. R. (2020). Authentication-based access control and data exchanging mechanism of IoT devices in fog computing environment. *Wireless Personal Communications*, 1–22.
54. Patonico, S., Braeken, A., & Steenhaut, K. (2019). Identity-based and anonymous key agreement protocol for fog computing resistant in the Canetti–Krawczyk security model. *Wireless Networks*, 1–13.
55. Pundir, S., Wazid, M., Singh, D. P., Das, A. K., Rodrigues, J. P. C., & Park, Y. (2020). Designing efficient sinkhole attack detection mechanism in edge-based IoT deployment. *Sensors, 20*(5).
56. Pundir, S., Wazid, M., Singh, D. P., Das, A. K., Rodrigues, J. J. P. C., & Park, Y. (2020). Intrusion detection protocols in wireless sensor networks integrated to internet of things deployment: Survey and future challenges. *IEEE Access, 8*, 3343–3363.
57. Rivest, R. L., Shamir, A., & Adleman, L. (1983). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 26*(1), 96–99.
58. Saha, S., Chattaraj, D., Bera, B., & Kumar Das, A. (2020). Consortium blockchain-enabled access control mechanism in edge computing based generic Internet of Things environment. *Transactions on Emerging Telecommunications Technologies*, e3995. https://doi.org/10.1002/ett.3995.
59. Saha, S., Sutrala, A. K., Das, A. K., Kumar, N., & Rodrigues, J. J. P. C. (2020). On the design of blockchain-based access control protocol for IoT-enabled healthcare applications. In *ICC 2020–2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland (pp. 1–6). https://doi.org/10.1109/ICC40277.2020.9148915.
60. Shabisha, P., Braeken, A., Kumar, P., & Steenhaut, K. (2019). Fog-orchestrated and server-controlled anonymous group authentication and key agreement. *IEEE Access, 7*, 150247–150261.
61. Singh, P., Nayyar, A., Kaur, A., & Ghosh, U. (2020). Blockchain and fog based architecture for internet of everything in smart cities. *Future Internet, 12*(4). https://doi.org/10.3390/fi12040061.
62. Singh, R., Tanwar, S., & Sharma, T. P. (2020). Utilization of blockchain for mitigating the distributed denial of service attacks. *Security and Privacy, 3*(3), e96. https://doi.org/10.1002/spy2.96.
63. Son, S., Lee, J., Kim, M., Yu, S., Das, A. K., & Park, Y. (2020). Design of secure authentication protocol for cloud-assisted telecare medical information system using blockchain. *IEEE Access, 8*, 192177–192191. https://doi.org/10.1109/ACCESS.2020.3032680.
64. Stallings, W. (2004). *Cryptography and network security: Principles and practices*, 3rd edn. India: Pearson Education.
65. Stojmenovic, I., Wen, S., Huang, X., & Luan, H. (2016). An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience, 28*(10), 2991–3005. https://doi.org/10.1002/cpe.3485.
66. Sutrala, A. K., Obaidat, M. S., Saha, S., Das, A. K., Alazab, M., & Park, Y. (2021). Authenticated key agreement scheme with user anonymity and untraceability for 5G-enabled softwarized industrial cyber-physical systems. *EEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2021.3056704.
67. Tutorial 3: Introduction to JSON-LD. (2017). http://www.linkeddatatools.com/introduction-json-ld. Accessed on February 2021.
68. Uriarte, R. B., & DeNicola, R. (2018). Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards. *IEEE Communications Standards Magazine, 2*(3), 22–28. https://doi.org/10.1109/MCOMSTD.2018.1800020.

69. Vangala, A., Bera, B., Saha, S., Das, A. K., Kumar, N., & Park, Y. H. (2020). Blockchain-enabled certificate-based authentication for vehicle accident detection and notification in intelligent transportation systems. *IEEE Sensors Journal*. https://doi.org/10.1109/JSEN.2020.3009382.
70. Vangala, A., Das, A. K., Kumar, N., & Alazab, M. (2020). Smart secure sensing for IoT-based agriculture: Blockchain perspective. *IEEE Sensors Journal*. https://doi.org/10.1109/JSEN.2020.3012294.
71. Vangala, A., Sutrala, A. K., Das, A. K., & Jo, M. (2021). Smart contract-based blockchain-envisioned authentication scheme for smart farming. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2021.3050676.
72. Wang, J., Wu, L., Choo, K. R., & He, D. (2020). Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure. *IEEE Transactions on Industrial Informatics, 16*(3), 1984–1992. https://doi.org/10.1109/TII.2019.2936278.
73. Wazid, M., Bagga, P., Das, A. K., Shetty, S., Rodrigues, J. J. P. C., & Park, Y. (2019). AKM-IoV: Authenticated key management protocol in fog computing-based internet of vehicles deployment. *IEEE Internet of Things Journal, 6*(5), 8804–8817.
74. Wazid, M., Bera, B., Mitra, A., Das, A. K., & Ali, R. (2020). Private blockchain-envisioned security framework for AI-enabled IoT-based drone-aided healthcare services. In *Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond (DroneCom'20)*, London, United Kingdom (pp. 37–42).
75. Wazid, M., & Das, A. K. (2016). An efficient hybrid anomaly detection scheme using K-means clustering for wireless sensor networks. *Wireless Personal Communications, 90*(4), 1971–2000.
76. Wazid, M., & Das, A. K. (2017). A secure group-based blackhole node detection scheme for hierarchical wireless sensor networks. *Wireless Personal Communications, 94*(3), 1165–1191.
77. Wazid, M., Das, A. K., Khan, M. K., Al-Ghaiheb, A. A., Kumar, N., & Vasilakos, A. V. (2017). Secure authentication scheme for medicine anti-counterfeiting system in IoT environment. *IEEE Internet of Things Journal, 4*(5), 1634–1646.
78. Wazid, M., Das, A. K., Kumar, N., Vasilakos, A. V., & Rodrigues, J. J. P. C. (2019). Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment. *IEEE Internet of Things Journal, 6*(2), 3572–3584.
79. Wazid, M., Das, A. K., Kumari, S., & Khan, M. K. (2016). Design of sinkhole node detection mechanism for hierarchical wireless sensor networks. *Security and Communication Networks, 9*(17), 4596–4614.
80. Wazid, M., Das, A. K., Shetty, S., & Jo, M. (2020). A tutorial and future research for building a blockchain-based secure communication scheme for internet of intelligent things. *IEEE Access, 8*, 88700–88716.
81. Wazid, M., Das, A. K., Shetty, S., Rodrigues, J. P. C., & Park, Y. (2019). LDAKM-EIoT: Lightweight device authentication and key management mechanism for edge-based IoT deployment. *Sensors, 19*(24). https://doi.org/10.3390/s19245539.
82. Wazid, M., Das, A. K., Shetty, S., & Rodrigues, J. J. P. C. (2020). On the design of secure communication framework for blockchain-based internet of intelligent battlefield things environment. In *IEEE INFOCOM 2020 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada (pp. 888–893). https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9163066.
83. Wu, D., & Ansari, N. (2020). A cooperative computing strategy for blockchain-secured fog computing. *IEEE Internet of Things Journal, 7*(7), 6603–6609. https://doi.org/10.1109/JIOT.2020.2974231.
84. Yang, C., Tan, L., Shi, N., Xu, B., Cao, Y., & Yu, K. (2020). AuthPrivacyChain: A blockchain-based access control framework with privacy protection in cloud. *IEEE Access, 8*, 70604–70615.
85. Zhang, L. (2019). Key management scheme for secure channel establishment in fog computing. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2019.2903254.