Design and Analysis of Access Control Mechanisms for IoT Applications using Blockchain-as-a-Service

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Sourav Saha

Roll No. 2019801003

sourav.saha@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA

APRIL 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this PhD thesis, titled "**Design and Analysis of Access Control Mechanisms for IoT Applications using Blockchain-as-a-Service**", by **SOURAV SAHA**, Roll No. 2019801003, has been carried out under my supervision and is not submitted elsewhere for a degree.

Place : IIIT Hyderabad

Date:

Adviser: **Dr. Ashok Kumar Das**
Professor
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, INDIA

Dedicated to the service of the Nation

Dissemination of Work

- **Chapter #4.**

  Basudeb Bera, Sourav Saha, Ashok Kumar Das, and Athanasios V. Vasilakos. "Designing Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System," in ***IEEE Internet of Things Journal***, Vol. 8, No. 7, pp. 5744-5761, April 2021, DOI: 10.1109/JIOT.2020.3030308. (2022 SCI Impact Factor: 10.6)

- **Chapter #5.**

  Sourav Saha, Basudeb Bera, Ashok Kumar Das, Neeraj Kumar, SK Hafizul Islam, and YoungHo Park. "Private Blockchain Envisioned Access Control System for Securing Industrial IoT-Based Pervasive Edge Computing," in ***IEEE Access***, Vol. 11, pp. 130206-130229, 2023, DOI: 10.1109/ACCESS.2023.3333441. (2022 SCI Impact Factor: 3.9)

- **Chapter #6.**

  Sourav Saha, Durbadal Chattaraj, Basudeb Bera, and Ashok Kumar Das. "Consortium blockchain-enabled access control mechanism in edge computing based generic IoT environment," in ***Transactions on Emerging Telecommunications Technologies (Wiley)***, Vol. 32, No. 6, pp. 1-34, Article No. e3995, 2021, DOI: 10.1002/ETT.3995. (2022 SCI Impact Factor: 3.6)

Acknowledgments

I would like to express my highest gratitude towards my supervisor Dr. Ashok Kumar Das. I am fortunate enough that I could work under his supervision. I thank for his immense support, invaluable supervision and incessant encouragement during my academic research. His tremendous experience and colossal knowledge consistently inspired me throughout my Ph.D journey and also transformed me as a better human being. Though words are not enough to express my respect towards him, but I sincerely respect and admire him equal to my parents. It is not possible to express his invaluable support that I can convey through words and I believe his blessing will be always there throughout my life.

I am fortunate and greatful to be a part of IIIT Hyderabad that provided me good research facilities and resources at the Center for Security, Theory and Algorithmic Research (CSTAR). I am thankful to Dr. Pawan Kumar (Assistant Professor, IIIT Hyderabad), Karthik Raj P, Krishna Kishore B, Srikanth K, and all professors and academic staffs of IIIT Hyderabad for their kind support and assistance. I am extremely grateful to thank the State and Central Governments of India for funding my entire education and fellowship throughout my study.

I would like to thank my friend Dr. Basudeb Bera (Research Fellow at the National University of Singapore, Singapore) for his extensive support in research and personal life. I am glad to thank my other friends: Ravi Kumar Thakur, Soumen Ghosh, Nilotpal Biswas, Ankush Mitra, Chandranath Biswas, and all my lab mates at IIIT Hyderabad. I also appreciate and thank to Dr. Neeraj Kumar (Professor, Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Punjab, India), Dr. Anil Kumar Sutrala (Principal Software Engineer at the CA Technologies – A Broadcom Company, Hyderabad, India), Dr. Mohammad Wazid (Professor, Graphic Era University, Dehradun, India), and Dr. Durbadal Chattaraj for their constructive support in research.

My utmost gratitude goes to my belated father. I am thankful to him for his great wisdom and unconditional love towards me. I am happy that I could endeavor to fulfill his wishes. I also thank to my mother and brother for their countless and unconditional love and faith.

Place: IIIT Hyderabad

Date:

Sourav Saha
PhD Student
Roll No. 2019801003
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, INDIA

Abstract

Blockchain is a distributed ledger and it validates the transactions without any intervention of a trusted third party (TTP). There are several advantages of using the blockchain-based smart grid infrastructure, because decentralization, immutability, transparency, confidentiality and trust are maintained. A blockchain-based smart grid system contains several entities, such as trusted registration authority (RA), service providers (SPs), IoT-enabled smart meters (SMs), and users associated with a smart meter. SPs organize the electricity allocation and energy trading system, and SMs are responsible for monitoring the power utilization and they maintain the pricing to the consumers (users). SMs can be deployed in the homes, and an attacker may capture some SMs and use the secure data stored into it. The communications between SPs and SMs must be secure so that passive/active attacks should not be possible. To ensure the security and privacy of the users' private information, it is extremely required to design a secure and efficient access control scheme between SMs and SPs. With the help from the blockchain technology, the secure data can be stored in the form of blocks in a private blockchain. The SPs involved in the P2P SP network are responsible in validating the new blocks before adding them into the blockchain using the consensus algorithm. To mitigate these issues, we first propose a new blockchain-based access control protocol in internet of Things (IoT)-enabled smart-grid system, called DBACP-IoTSG. Through the proposed DBACP-IoTSG, the data is securely brought to the service providers from their respectively smart meters. The Peer-to-Peer (P2P) network is formed by the participating services providers, where the peer nodes are responsible for creating the blocks from the gathered data securely coming from their corresponding smart meters and adding them into the blockchain after validation of the blocks using the voting-based consensus algorithm. In our work, the blockchain is considered as private because the data collected from the consumers of the smart meters are private and confidential. By the formal security analysis under the random oracle model, non-mathematical security analysis and software-based formal security verification, DBACP-IoTSG is shown to be resistant against various attacks. We carry out the experimental results of various cryptographic primitives that are needed for comparative analysis using the widely-used Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL). A detailed comparative study reveals that DBACP-IoTSG supports more functionality features and provides better security apart from its low communication and computation costs as compared to recently proposed relevant schemes. In addition, the blockchain implementation of DBACP-IoTSG has been performed to measure computational time needed for the varied number of blocks addition and transactions per block in the blockchain.

The Industrial Internet of Things (IIoT) is able to connect machines, analytics and people with IoT smart devices, gateway nodes and edge devices to create powerful intuitivenesses to drive smarter, faster and effective business agreements. IIoT having interconnected machines along with devices can monitor, gather, exchange, and analyze information. Since the communication among the entities in IIoT environment takes place insecurely (for instance, wireless communications and Internet), an intruder can easily tamper with the data. Moreover, physical theft of IoT smart devices provides an intruder to mount impersonation and other attacks. To handle such critical issues, we next design a new private blockchain-envisioned access control scheme for Pervasive Edge Computing (PEC) in IIoT environment, called PBACS-PECIIoT. We consider the private blockchain consisting of the transactions and registration credentials of the entities related to IIoT, because the information is strictly confidential and private. The security of PBACS-PECIIoT is significantly improved due to usage of blockchain as immutability, transparency and decentralization along with protection of various potential attacks. A meticulous comparative analysis exhibits that PBACS-PECIIoT achieves greater security and more functionality features, and requires low costs for communication and computational as compared to other pertinent schemes.

The communication among various entities in an edge computing based generic IoT environment takes place via insecure (public) channel (e.g., via the Internet). It gives an opportunity to an adversary to mount various types of attacks, including "replay", "man-in-the-middle", "impersonation", and "Ephemeral Secret Leakage (ESL)" attacks. Moreover, the adversary can physically capture to some IoT smart devices in order to compromise secure communication among other non-compromised nodes in the network. Therefore, it is very much essential that the data needs to be securely exchanged among various entities in the network so that it will not be revealed to the adversary. Otherwise, if the adversary can manipulate the genuine information, the transactions contained in a block in the blockchain will not be also genuine. The access control mechanism plays a very crucial role here, because the IoT devices require to send the information to their nearby gateway node and also the gateway node needs to send the data to its associated edge server(s) securely. As a result, we need to have a secure access control mechanism in edge computing based generic IoT environment to make secure communication among various entities in the network. The edge servers in a peer-to-peer (P2P) edge servers network are considered as trusted and they are responsible for creating, verifying and adding the blocks in their local ledgers first using consensus algorithm. Later, the local ledgers maintained by the edge servers are then periodically updated in the blockchain's global ledger in order to avoid much burden at the blockchain

center. Due to blockchain technology, once the blocks are added into the blockchain, these can not be further modified, updated or even deleted from the blockchain because each block contains previous block hash, Merkle tree root, signature on the transactions and also current block hash. Most of the access control protocols proposed in the literature in the IoT and also in resource-constrained wireless sensor networks environments are vulnerable to attacks, and they do not support blockchain feature in order to provide stronger security and more functionality features, such as block verification in blockchain, transparency, decentralization and immutability properties. Finally, we design a novel access control protocol in edge computing based generic IoT environment where depending on the importance of the data in IoT applications, the data are encrypted in the block (private blockchain) or these are stored in unencrypted form in the block in the blockchain (public blockchain). There may be some applications where we need to have both private and public data to be stored in a block in the blockchain (consortium blockchain). Hence, we consider consortium blockchain access control mechanism to address these issues. Towards this, a new consortium blockchain-enabled access control scheme in edge computing based generic IoT environment (called CBACS-EIoT) has been suggested, where the mutual authentication among the IoT smart devices and the gateway node(s), and also among the gateway node(s) and respective edge server(s) occur. In addition, key management phase is executed among the edge server(s) and associated cloud server(s). Using the established secret keys, the entities in the network communicate securely. The data gathered securely by the gateway nodes is then used to form various types of blocks (private, public or consortium) at the edge server(s) based on application types in the generic IoT environment. The created blocks are mined by the edge servers in order to add them in the blockchain center. A detailed security analysis including the formal security and also the simulation based formal security verification on CBACS-EIoT have been carried out to exhibit that CBACS-EIoT is secure against passive and active attacks. Finally, a meticulous comparative performance analysis shows that CBACS-EIoT offers superior security and supports more functionality features, and also provides less communication and computational overheads as compared to existing relevant schemes.

Keywords: Internet of Things (IoT), Industrial Internet of Things (IoT), smart grids, edge computing, blockchain, access control, authentication and key agreement, session key, formal security, formal security verification, distributed system, peer-to-peer network, consensus algorithms, Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL), Automated Validation of Internet Security Protocols and Applications (AVISPA).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An Internet of Things (IoT) consists of a huge number of things (devices) that are interconnected through the Internet. A "thing" or "object" in an IoT environment may be a person, animal or physical object that can be assigned a unique identifier (IP address or device ID). IoT devices can transfer sensing information from their surrounding areas via the Internet to some server(s) for further processing. IoT devices can conduct several tasks like "remote sensing", "actuating (making an action)" and "support monitoring capabilities". Such devices can be made smart appropriately so that they can operate without any human intervention. Thus, the objective of IoT is to offer a strong interaction among the physical world and computer-based systems that can lead to improvements in the economic welfare, along with the accuracy and efficiency while minimizing human interventions.

Figure 1.1 shows a generic IoT network architecture mentioning four different scenarios (e.g., home, transport, community and national). Several smart devices, like sensors and actuators can be installed or deployed in various applications where they are connected to the Internet via the trusted Gateway Nodes (GWNs). The information (data) accessed by the IoT devices can be accessed by various users (i.e., in case of "a smart home user in a home application" and in case of "a doctor in a healthcare application") [85].

Figure 1.2 also illustrates a generic IoT-based smart home application as a case study [187]. The smart devices are deployed into two groups: a) appliance and b) monitor. The devices installed in the appliance and monitor groups, known as the "agents", and communicate with the central controller via wireless communications. A user can control a smart home system by using the user interface. Moreover, the data gathered by any IoT smart device in the monitoring group can be also accessed by a user.

Gartner Inc. [2] forecasted that the number of connected IoT devices can reach to 20.4

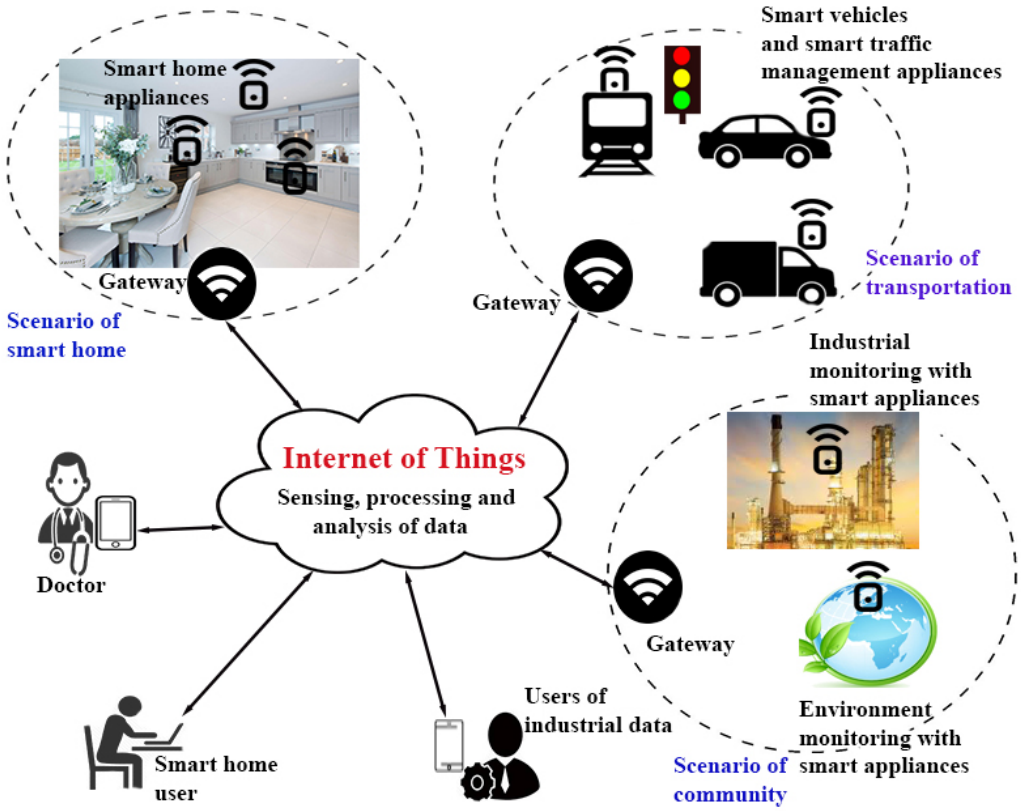

Figure 1.1: IoT environment with various applications [214, 218]

Table 1.1: Various IoT units installed based by category (millions of units) [2]

Category	2016	2017	2018	2020
Consumer	3,963.00	5,244.30	7,036.30	12,863.00
Business: cross-industry	1,102.10	1,501	2,132.60	4,381.40
Business: vertical-specific	1,316.60	1,635.40	2,027.70	3,171
Total	6,381.80	8,380.60	11,196.60	20,415.40

billion by the year 2020. A summary of the number of IoT units (grouped by category) installed in terms of millions of units in 2016-2017 and the predicted number of units in 2018 and 2020, is illustrated in Table 1.1. IoT applications for smart TVs and digital set-top boxes are being utilized by consumers, whereas the smart electric meters or smart meters and commercial security cameras are being broadly used in smart grid implementations and businesses, respectively [2]. Moreover, the industrial IoT applications and devices, such as

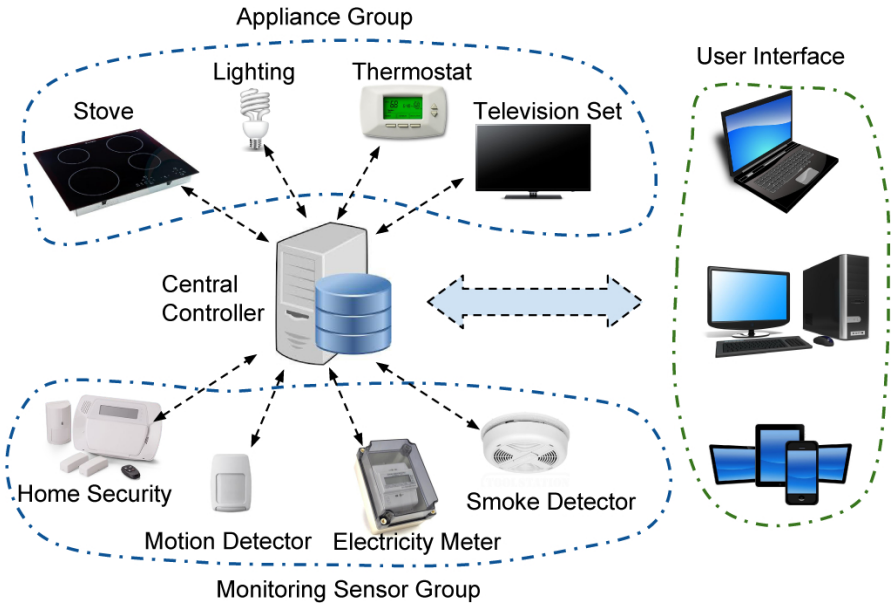

Figure 1.2: An example of an IoT-based smart home application [187]

manufacturing field devices, real-time location devices for healthcare applications and sensors for electrical generating plants are the connected things in different businesses and manufacturing plants. It was pointed out that by the year 2020, cross-industry devices may reach 4.4 billion units, while the vertical-specific devices (e.g., specialized equipment used in hospital operating theaters and tracking devices in container ships) was expected to reach 3.2 billion units.

Table 1.2: IoT endpoint spending by category (millions of dollars) [2]

Category	**2016**	**2017**	**2018**	**2020**
Consumer	532,515	725,696	985,384	1,494,466
Business: cross-industry	212,069	280,059	372,989	567,659
Business: vertical-specific	634,921	683,817	736,543	863,662
Total	1379,505	1,689,572	2,094,881	2,925,787

Since the consumers are supposed to buy more IoT devices in the future, business investments are more likely to increase in future years. The IoT endpoint spending by category (millions of dollars) [2] is briefed in Table 1.2.

In recent years, the growth of IoT has lead to creation of a huge volume of data that

demands "massive computing resources, storage space and communication bandwidth" [151]. According to the prediction made by Cisco, 50 billion IoT devices are expected to be connected to the Internet by 2020 [74], and this number may hit to 500 billion by 2025 [35]. It is also predicted that the "data produced by human, machines and *things* would reach 500 zettabytes by 2019, but the Internet Protocol (IP) traffic of global data centers would only reach 10.4 zettabytes" [3]. Hence, based on the observation reported in [36], "45% of IoT-created data would be stored, processed and analyzed upon close to, or at the edge of network".

1.1 IoT applications

There are various applications of a generic IoT network which are discussed as follows.

1) IoT based healthcare system

IoT-enabled healthcare system improves the quality of health services and provide better healthcare facilities to every individuals across the world. In healthcare system a smart sensors are connected with the network and continuously monitor the patient also constantly send and receive health data with the hospital server. Moreover, the smart sensors can also able to send emergency response to the concern authority. In the healthcare application different IoT devices (also known as smart devices) are attached within the body of a person, and these deployed devices are also called as "wireless body area network (WBAN)" [189]. The devices are connected with the network and communicate each other to continuously monitor the patient and the data are gathered by the mobile device and sent to the concern server via access points. Moreover, it helps people who lives in the remote locations where in case of critical scenario, hospitals can suggest to take appropriate actions to save the life of a patient. However, as the data are strictly confidential and private, the data in transit must be not leaked to any situation to the adversary [136].

There are many security issues raises, where if an unauthorized person gets access, it may lead to catastrophic of a patient. Access control is one of the important security feature that is essential for an IoT-enabled healthcare system that authorized user using his/her smart mobile device to authenticate with the trusted Hospital Authority (HA) in a hospital [161]. Moreover, in situation like Coronavirus disease (COVID-19), many people carry out their COVID tests from various diagnostics centers and if the patient's medical tests are critical or serious, the doctor may advised the patient to go for hospitalization [25, 40, 53]. Moreover, the patient may choose different hospital for his/her treatment and the patient also need to

share the health data with other healthcare providers where the medical tests has been done. Therefore, the security and privacy of patient's data is very important to share with other hospitals. In such scenarios, it is required and necessity to design an appropriate access control mechanism for such applications.

2) Industrial IoT system

In Industrial IoT (IIoT) system, sensors/actuators collect data from various machines in which machines can become more consistent and accurate, and produce more efficiency in the production. At the same time, it is also needed to achieve the quality control and sustainability in industrial IoT system. In smart transportation application, IoT devices monitor various parameters, such as traffic congestion, pollution in the air and surveillance, in order to make the system automatic. One of the challenges faced in this application is how to ensure secure data collection [59].

The recent development in IIoT systems it provides highly automated, dynamic and smart machines which communicates with each other to produce maximum productivity, efficiency, product quality and reduce product cost compared to traditional industry. Moreover, IIoT system produces lot of data and it is a challenging task to protect individual data [113]. In various cases it has been found that there are huge leakage in IIoT such as industrial power consumption, raw material information, air quality, product planning, reactivity of a catalyst in different temperature and air pressure conditions, and so on. An adversary may infer such data and leakage of such information may reduce the productivity and efficiency of the IIoT system. Therefore, it is important to design secure access control system between the communicated devices [206].

3) Other potential applications

Some other potential applications that use the IoT as a technology are as follows:

- **Smart home**: In smart home application, IoT based smart homes can provide controlling and customizing the home appliances remotely. Though it has numerous benefits, but also have several security challenges, such as privacy, integrity, confidentiality and authenticity. The smart devices are connected with their nearby gateway node(s) via the public Internet, and the gateway nodes forward the data to the nearby edge servers in case of an edge-based IoT environment [219].

- **Internet of Vehicles (IoV) and smart transportation**: IoV has numerous potential applications which comes with the use of the IoT-empowered smart devices. In IoV, vehicles, roads, street signs and traffic lights can accordingly adjust to changing conditions in order to assist the drivers, which will provide improve safety, ease congestion and pollution reduction in the transportation system. Public communication among various involved entities in the IoV environment makes several vulnerabilities and an adversary can tamper with the communicated data also the adversary may launch several attacks [209]. Therefore, it is required to provide better security and privacy of the communicated data in IoV system over the public domain.

- **Smart grid**: The popularity and demand of the IoT-based smart grid is growing rapidly. The IoT-based smart grid is embedded with smart sensors, objects, and actuators (IoT smart devices). It provide a reliable transmission of energy which makes the system automated and efficient, and also improves the economic benefits and deliver quality of services. The smart objects or physical devices are integrated with an interconnected network that helps in delivering the services efficiently. It has numerous advantages, but at the same time it has also numerous challenges, such as centralized control, transparency, poor interoperability, and privacy and security issues, including energy trading between untrusted/nontransparent networks, auditing and verifying of transaction records in the system [113].

- **Internet of Drones (IoD)**: Due to increase of commercial drones applications, recent forecasts indicate that there will be a 100 USD billion market opportunity over the coming years based on the drones. A drone can be equipped with various IoT devices, such as cameras, thermal infrared imaging, Global Positioning System (GPS), and various other sensors to detect sound and image of a particular location. There are several applications of drones where the drones can be widely used, ranging from military, news-gathering (for example, videography and photography), security, agricultural and logistics deployments, surveillance, medicine to traffic-monitoring applications. In IoD system sensors are attached with the drones and communications happen over the public channel, it faces various threats such as attacks during communication between drone to drone or drone to ground server station, physical capture of some drones and captured credentials stored within it using power analysis attacks [188].

- **Smart agriculture**: There are various fields where IoT plays a mojor role in smart agriculture application. In agriculture monitoring, it monitors the affect and growth of

the plants in every stage for their growth. However, various sensors are used to monitor and gathering knowledge for agriculture, water, soil, plant monitoring, and irrigation control and so on.

1.2 Advantages of IoT

There are various advantages and benefits of IoT applications, and some of them are listed below.

- *Cost effective:* IoT helps to monitor the real time data with cost effectively and nominal disturbance, such as wildlife monitoring, flooding and military surveillance, fire and crops monitoring, and so on.
- *Time efficient:* In various applications such as medicine delivery, food or package delivery, traffic monitoring it reduces the time and provide a qucker services to the consumers.
- *Reducing life threatening risks:* With the help of IoT application, a patient can be monitor remotely in case of major heart attacks or other health disease the IoT sensors can sent alert to hospital and the patent can be moved to a hospital on prior time.
- *Transformation of smart cities:* Using IoT system life in metropolitan cities become easy by providing services like smart parking and better navigation, real time view of the traffic, accident alert and route optimization.
- *Precision:* A Global Positioning System (GPS) is installed in drone system to monitor or control for a specific locations to observe farming obligations such as pesticide spraying, identification of weeds, monitoring crop health, crop damage, crop assessment, field soil analysis, irrigation monitoring and so on.

1.3 A generic architecture of an IoT system

The basic architecture of IoT consists of different IoT applications, such as smart home, healthcare, industrial monitoring, smart vehicles and smart traffic management appliances, etc. For each application, there are various component such as trusted registration authority (RA), several IoT smart devices (SD_i) are that installed in a proximity of their associated gateway node (GW_j). One or mode GW_j can be associated with their nearby edge server ES_l.

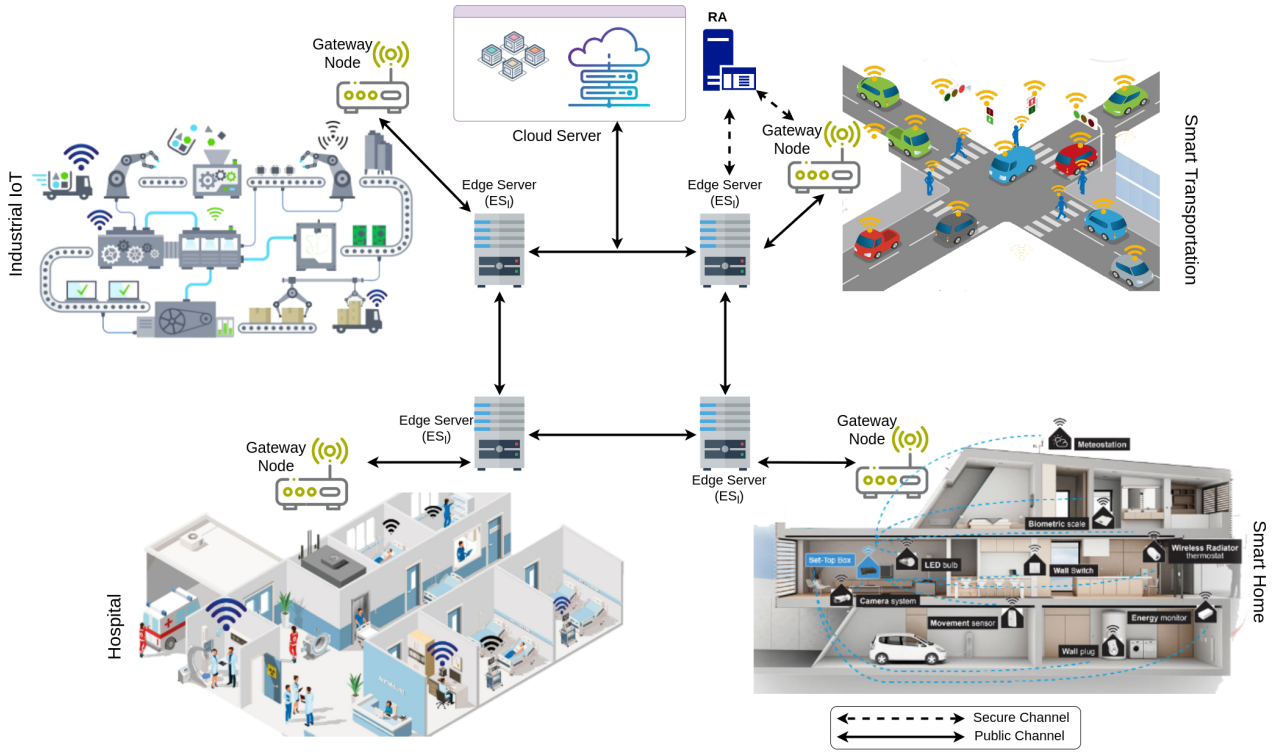

Figure 1.3: A generic architecture of an IoT environment

The responsibility of *RA* is to register all the entities before starting any communication. Figure 1.3 shows a generic architecture for IoT environment. A group of authorized edge servers form a peer-to-peer (P2P) edge servers network, called P2P ES network. The peers in the P2P ES network are responsible for collecting the data securely, validate it and process to the cloud server for storage.

Each component in the IoT application is described below.

- *IoT smart device:* Several IoT smart devices can be installed or deployed in a particular application. The smart devices are typically responsible for gathering their surrounding information and transmit them to their nearby gateway nodes for further processing. The IoT smart devices are typically resource limited in terms of memory, storage, communication and computation capabilities.

- *Gateway node:* A gateway node (GW) acts as an access point for a particular application. The data collected by the GW form various smart IoT devices, which are then forwarded to its associated edge sever (ES) for further processing.

- *Edge server:* An edge server is a device that manages the flow of data at the boundary

among the network. An edge server contains various roles that are dependent on what kind of devices used in the particular application. It typically acts as a network entry (or exit) point, and is also responsible for various activities, such as "transmission", "processing", "routing", "filtering", "monitoring", "translation", "storage of information passing between networks", etc.

- *Registration authority:* A registration authority (RA) is basically a fully trusted entity in the network, which is responsible for registering all the deployed IoT smart devices, the gateway nodes, edge servers and cloud servers in the BC. After successful registration, each entity is pre-loaded with proper credentials prior to its deployment or placement in the edge-based IoT environment.

1.4 Motivation and objective of the work

In Industrial Internet of Things (IIoT) environment, there are various types of applications connected with the system and they integrate large-scale discrete heterogeneous data. Such data can be from the smart sensor data, health care data, traffic data, environmental monitoring data, and industrial manufacturing data. In some smart energy industries, sensors, machines, and actuators collect huge amount of data such as energy, air quality, fault and resource prediction, and product planning from various locations. It further produces large data and enforce a huge amount of processing time to store the data in traditional centralized system. Moreover, in chemical industry, there is an extensive amount of critical data, such as reactivity of a catalyst in different temperature and air pressure conditions, results after the chemicals reactions. In such scenario, inefficiency of IIoT system can seriously damage the productivity of the industry.

Blockchain is a distributed chain of structure on a decentralized Peer-to-Peer (P2P) network, which eliminates the requirement of centrally controlled system and allows the network entities to store the data in a distributed fashion. Considering the demand of large-scale IoT systems, it becomes infeasible and inefficient to store the huge volumes of data in a traditional IoT system. Therefore, we feel that a great essence in designing blockchain-based IoT systems. Thus, it should provide an efficient and robust solution to deal with the security requirements needed for IoT environments. Moreover, due to wireless communication happen among different entities in IoT, an adversary should not be able to tamper with the sensitive data. Tampering of data may include intercepting, modifying, deleting or even inserting fake information during communication.

With the help of the cloud computing upgradation, IoT platform can process information in a traditional manner and transform the information into the real time actions. While the cloud storage becomes an important role in an IoT or IIoT environment, however there are issues related to threat of data, transparency and privacy preservation. This demands that we require to integrate the blockchain technology with the industrial IoT applications. Since the blockchain helps in providing the trusted sharing services where the reliable information and data can be retrieved, the data (information) can be then traceable. At the same time, the blockchain is also immutable; thus it enhances the security as well. Therefore, integration of decentralized blockchain in IIoT system can enable better efficiency, transparency and guarantee security solutions.

In an IoT environment, some smart devices may be physically captured or their battery may drain out. Therefore, to prolong the lifetime of smart devices in the IoT environment, new smart devices deployment is an important task. The deployed smart devices may not be always assumed to be genuine ones as an adversary may deploy some malicious smart devices in the network. In this case, it becomes hard to distinguish malicious new nodes from genuine nodes in the IoT environment. This requires an access control method when new sensing nodes are deployed in order to prevent malicious nodes from entering the network. Such an access control method primarily deals with the following two tasks:

- *Node authentication:* The newly deployed sensing node must authenticate itself to its neighbor nodes to prove that it is a legal node for accessing the information from the other sensing nodes.

- *Key establishment:* A newly deployed sensing node should create a shared secret keys with its existing neighbor nodes in order to ensure secure communication during the transmission of information.

Additionally, a new sensing node's addition phase is required for access control mechanism in the IoT environment. Access control mechanisms are classified into two categories based on their authentication type: certificate-less and certificate-based.

Most of the existing access control schemes proposed in the literature for both IoT and its related environments, such as in wireless sensor networks (WSNs), smart grids, smart homes, and healthcare, are either insecure against various known attacks or they are inefficient in communication as well computation. To mitigate these issues, in this thesis, we aim to design novel secure access control schemes for IoT environment using the blockchain technology in order to eradicate the security pitfalls in the existing device access mechanisms.

1.5 Summary of contributions

The reserach contributions towards this thesis have been summarized as follows.

1.5.1 Contribution #1

In the first contribution of the thesis, we propose a novel decentralized private blockchain-based access control protocol in IoT-enabled smart-grid system, called DBACP-IoTSG. In DBACP-IoTSG, registration of smart meters and service providers is performed in offline mode prior to deployment in the IoT-enabled smart-grid environment. The access control phase associated with DBACP-IoTSG permits a smart meter, say SM_i to mutually authenticate with its associated service provider, say SP_j through "node authentication" process and then to establish a session key among them for secret communication through "key establishment" process. The pairwise secret keys among the service providers are used for secure consensus procedure. DBACP-IoTSG also supports dynamic node addition mechanism. We then provide a detailed mechanism for a new block creation and addition in the blockchain through the "Practical Byzantine Fault Tolerance (PBFT)" based consensus mechanism [38]. The proposed DBACP-IoTSG allows secure leader selection in the P2P SP network that is responsible for a block verification and addition in the blockchain using voting-based PBFT consensus algorithm.

The proposed DBACP-IoTSG allows to preserve both anonymity and untraceability properties that are extremely needed in an IoT-enabled smart grid environment. In addition, DBACP-IoTSG also permits dynamic smart meter addition phase after initial deployment in an event that if some service providers SP_j may become faulty nodes or some smart meters SM_i may be physically compromised by an adversary. Furthermore, DBACP-IoTSG resists a crucial active attack under the present *de facto* model, known as the Canetti and Krawczyk's model (CK-adversary model) [37], known as "ephemeral secret leakage (ESL)" attack. Next, we provide a rigorous security analysis through the formal security under the broadly-accepted "Real-Or-Random (ROR) oracle model" [37], informal (non-mathematical) security analysis and also formal security verification using the widely-used AVISPA automated software tool [17] through simulation.

We also provide the experimental results of various cryptographic primitives that are needed for comparative analysis using the widely-used "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5]. A detailed comparative analysis among DBACP-IoTSG and other relevant protocols in smart grid environment shows that

DBACP-IoTSG supports better security and provides more functionality attributes, and also requires less communication and computation costs. Finally, the blockchain implementation of DBACP-IoTSG has been carried out in order to measure computational time required for the varied number of blocks addition as well as the varied number of transactions per block in the blockchain.

1.5.2 Contribution #2

In the second contribution of the thesis, we propose a novel "private blockchain-envisioned access control scheme for Pervasive Edge Computing (PEC) in IIoT environment, called PBACS-PECIIoT". The purpose behind applying the the private blockchain is that the transactions and registration credentials of the entities related to IIoT are confidential and private. In the proposed PBACS-PECIIoT, registration credentials obtained by a smart device (SD_i) and the gateway node (GN_j) are fetched from the Blockchain during the access control phase for authentication and key agreement purposes. Additionally, it is also worth to notice that the registration credentials stored in the blockchain center (BC) are fetched by an edge server for key management purpose with its gateway node. The proposed scheme makes use of the blockchain during the registration as well as access control and key management phases which is significantly different from the first contribution.

After collecting the information securely from the deployed IoT smart devices by their respective gateway node(s), the information is securely delivered to the edge servers by their associated gateway nodes in form of transactions. The edge servers are then responsible for building the blocks, verifying and adding them in the private blockchain with the help of the proposed voting-based PBFT algorithm. The local ledgers are maintained by the edge servers in the blockchain center. A detailed security analysis including the formal security verification has been conducted. It demonstrates that PBACS-PECIIoT is secure against a number of potential attacks against passive/active adversaries. The "real testbed experiments for various cryptographic primitives with the help of widely-accepted Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5] have been performed under both server and Raspberry PI 3 platforms. These testbed experiments measure the computational time for the primitives with respect to these platforms. Moreover, a detailed comparative analysis among PBACS-PECIIoT and other related existing schemes has been performed. It shows the effectiveness and robustness of PBACS-PECIIoT over other schemes. The proposed PBACS-PECIIoT is also implemented through blockchain simulation study in order to measure its performance as well as computational time.

1.5.3 Contribution #3

In the last but not the least contribution of the thesis, a novel access control protocol in edge computing based generic IoT environment has been designed (called CBACS-EIoT), where depending on the importance of the data in IoT applications, the data are encrypted in the block (private blockchain) or these are stored in unencrypted form in the block in the blockchain (public blockchain). There may be some applications where we need to have both private and public data to be stored in a block in the blockchain (consortium blockchain), for example, a smart transportation system.

CBACS-EIoT offers access control among IoT smart devices and their associated gateway nodes and also among the gateway nodes and edge servers. In addition, key management process among the edge servers and the cloud servers in the blockchain center. The blocks created by the edge nodes are mined and put in their respective local ledgers. The local ledger having blocks are then added in the global ledger maintained by the cloud servers in the blockchain center. All the secure communications among the IoT smart devices, gateway nodes, edge servers and cloud servers happen using their respective secret (session) keys.

To assure that the proposed CBACS-EIoT is highly secure against various potential attacks needed for an IoT environment, the formal security and informal security analysis, and also the formal security verification using the broadly accepted AVISPA tool have been performed. A meticulous comparative analysis on security and functionality features, communication and computation overheads among the proposed CBACS-EIoT and existing schemes has been also performed to demonstrate the superiority of security and efficiency of CBACS-EIoT over other existing schemes.

1.6 Organization of the thesis

The organization of this thesis is as follows.

- **Chapter** 1 gives a brief overview of IoT architecture and its various applications, security and functionality requirements. It also discusses a taxonomy of various security protocols in IoT. We then focus on access control, and the objective and motivation behind the research work on access control mechanisms proposed for IoT using the blockchain technology as a service.

- **Chapter** 2 discusses some mathematical preliminaries that are useful for discussing the proposed schemes in this thesis work.

- **Chapter** 3 presents the existing related work on various access control mechanisms in IoT-related environment using both the blockchain and non-blochain services.
- In **Chapter** 4, we propose a novel decentralized private blockchain-based access control protocol in IoT-enabled smart-grid system, called DBACP-IoTSG.
- In **Chapter** 5, we design a novel private blockchain-envisioned access control scheme for Pervasive Edge Computing (PEC) in IIoT environment, called PBACS-PECIIoT.
- In **Chapter** 6, we then propose a novel access control protocol in edge computing based generic IoT environment (called CBACS-EIoT) using consortium blockchain as a service.
- Finally, **Chapter** 7 summarizes the thesis by highlighting the research contributions and then providing some future research directions.

Chapter 2

Mathematical Preliminaries

In this chapter, we explain some mathematical preliminaries which are required to design various access control protocols in an IoT environment. We first start with the discussion on one-way cryptographic hash function, its properties, and different associated applications. Next, we discuss about the elliptic curve and its properties followed by elliptic curve cryptography (ECC) and its various computationally hard problems. We discuss about an automated simulation tool (known as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [17]) for checking whether a designed security protocol is safe under the Dolev-Yao (DY) threat model [67] or not. Finally, we discuss the blockchain technology, its various applications and underlying consensus protocols, and the Merkle tree that will be useful for block construction in the blockchain.

2.1 Cryptographic one-way hash function

A one-way function is defined as a function for which finding the inverse of any random input is computationally infeasible. A hash function is one that produces a fixed length output for any arbitrary length input. In cryptography, a one-way hash function is used to produce a digest or a hash value of a message with the following properties:

- The output is deterministic, that is, the same digest is produced for the same message.
- If the input message is altered even slightly, the hash digest should change significantly to reduce the probability of correlation between the two hash values.
- Deriving the input x from the given hash value $y = h(x)$ and the given hash function $h(\cdot)$ is computationally infeasible. This property is called the *one-way* property.

- For any input x, finding another input y such that $h(x) = h(y)$ with $y \neq x$, is computationally infeasible. This property is otherwise known as the *weak collision resistant* property.
- Identifying an input pair (x, y) such that $h(x) = h(y)$ where $y \neq x$, is also computationally infeasible. This property is otherwise known as the *strong collision resistant* property.

Mathematically, a one-way hash function can be defined as follows.

Definition 2.1 (One-way hash function)**.** *A "collision-resistant cryptographic one-way hash function" h: $\{0,1\}^* \rightarrow \{0,1\}^l$ is treated as a "deterministic function that on a variable length input string produces a fixed length output string of l bits, say". Let $Adv_{(A)}^{HASH}(rt)$ denote the "advantage of an adversary $\mathcal{A}$ in finding a hash collision" in $h(\cdot)$. Then, $Adv_{(A)}^{HASH}(rt) = Pr[(i_1, i_2) \in_R \mathcal{A}: i_1 \neq i_2, h(i_1) = h(i_2)]$, where $Pr[E]$ is the "probability of a random event E", and the input pair $(i_1, i_2) \in_R \mathcal{A}$ means that the input strings i_1 and i_2 will be randomly picked by $\mathcal{A}$. We say "an (ζ, rt)-adversary $\mathcal{A}$ attacking the collision resistance of $h(\cdot)$" means that $\mathcal{A}$'s runtime is at the most rt and that $Adv_{(A)}^{HASH}(rt) \leq \zeta$.*

2.1.1 Various hash functions

In the following, we discuss some widely-used hash functions.

- **Message Digest (MD) family**: This hash family carries a variant of hash functions, such as MD2, MD4, MD5, and MD6. MD2 is designed by Ronald Rivest in 1989 which takes an arbitrary length input and produces an 128-bit length output, and it is reported in Request for Comments (RFC) 1115. MD2 is considered no longer secure one-way hash function as it had preimage attack found in 2008, and collision attack was revealed in 2009. Next, MD4 was developed by Ronald Rivest in 1990 [170]. It takes an arbitrary length input and produces an 128-bits output. The security of MD4 has been harshly compromised, and the first collision attack was reported in 1995. Then MD5 was introduced by Ronald Rivest in 1992 which produces an 128-bits string as the output. It is also cryptographically broken, but it is still widely used as a checksum to verify data integrity against unintentional corruption. After that, MD6 was published in 2008 as a cryptographic hash function and it was also submitted to the National Institute of Standards and Technology (NIST) Secure Hash Algorithm (SHA)-3 competition. A short description of the MD family is shown in Table 2.1.

Table 2.1: MD variants and their properties

MD variant	Input size (in bits)	Output size (in bits)	Block size (in bits)	No. of rounds	Publication year	Designer(s)
MD2	Arbitrary	128	128	18	1989	Ronald Rivest
MD4	Arbitrary	128	512	3	1990	Ronald Rivest
MD5	Arbitrary	128	512	64	1992	Ronald Rivest
MD6	Arbitrary	Variable ($0 < d \leq 512$)	Variable	–	2008	Ronald Rivest and others

- **Secure Hash Algorithm (SHA)**: This hash family carries various secure hash functions, such as SHA-1, SHA-2, and SHA-3, whereas SHA-2 has the variants of SHA-226, SHA-256, SHA-384 and SHA-512, and SHA-3 also has "SHA3-224", "SHA3-256", "SHA3-384", and "SHA3-512" types of variants [140]. SHA-1 was introduced in 1995, which takes an arbitrary length (in bits) input size and make blocks of 512-bits blocks and produces 160-bit output after 80 rounds. It was developed by the United States "National Security Agency (NSA)", and is "U.S. Federal Information Processing Standard". The original document of this algorithm (SHA-0) was published by National Institute of Standards and Technology (NIST) in 1993 under the Secure Hash Standard, Federal Information Processing Standards Publications (FIPS PUB) 180. Later, the algorithm was developed with various variants and they are mention in Table 2.2. Nowadays, SHA-256 is used in various applications, like ESDSA, blockchain technology and so on.

Table 2.2: SHA variants and their attributes

SHA variant	Message size (in bits)	Block size (in bits)	Output size (in bits)	No. of rounds	Publication year	Designer(s)
SHA-1	$< 2^{64}$	512	160	80	1995	NSA
SHA-224	$< 2^{64}$	512	224	64	2001	NSA
SHA-256	$< 2^{64}$	512	256	64	2001	NSA
SHA-384/SHA-512	$< 2^{128}$	1024	384/512	80	2001	NSA

- **RIPEMD**: Hans *et al.* [66] designed a "fast cryptographic hash function", called RIPEMD-160. It was developed in the framework of the "EU project RIPE (Race Integrity Primitives Evaluation)". As the name suggests, this hash function produces 160-bit hash output as a message digest. RIPEMD-160 was designed towards the software implementations that can be done easily on 32-bit architectures. Given an arbitrary size input message (string), RIPEMD-160 compresses the input by dividing it into the blocks, whose size is 512 bits each, as in SHA-1.

2.1.2 Cryptographic applications of hash functions

Hash functions can be used in building other cryptographic primitives like message authentication codes and pseudo-random number generators. They also help in verifying the integrity of a message. Since it is very sensitive to even a small variation in input, hash digest can be used to avoid storing passwords in cleartext. The Secure Hash Algorithm (SHA) standard has algorithms with varying lengths of digest produced. Of these, the SHA-1 [2] with a 160-bit hash digest is the most widely used in applications and protocols like Secure Socket Layer (SSL). For better security, SHA-256 is preferred.

2.2 Elliptic curve and its properties

Consider a set $E_p(a, b)$ of all solutions $(x, y) \in Z_p \times Z_p$ corresponding to a non-singular elliptic curve of the form:

$$y^2 = x^3 + ax + b \pmod{p}$$

over a prime or Galois field $GF(p)$, where $p > 3$ is prime number, $a, b \in Z_p$ are two constants fulfilling the condition $4a^3 + 27b^2 \neq 0 \pmod{p}$, and $Z_p = \{0, 1, \ldots, p-1\}$. Let $\mathcal{O}$ denote the the point at infinity or zero point in $E_p(a, b)$. Then, $E_p(a, b)$ constitutes an abelian or commutative group with respect to addition modulo p operation with $\mathcal{O}$ as the additive identity.

The condition $4a^3 + 27b^2 \neq 0 \pmod{p}$ is the necessary and sufficient condition that the corresponding elliptic curve has a non-singular solution [153]. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be two points in $E_p(a, b)$, where x_P and y_P denote the x and y co-ordinates of the point P, respectively, and x_Q and y_Q denote the x and y co-ordinates of the point Q, respectively. Then, if $P + Q = \mathcal{O}$, then $x_Q = x_P$ and $y_Q = -y_P$. In addition, $P + \mathcal{O} = \mathcal{O} + P = P$, $\forall P \in E_p(a, b)$, where $\mathcal{O}$ is known as the additive identity. More precisely, the number of

points on $E_p(a,b)$, which is denoted by $\#(E)$, satisfies the following inequality [192]:

$$p+1-2\sqrt{p} \leq \#(E) \leq p+1+2\sqrt{p}.$$

In other words, the elliptic curve $E_p(a,b)$ over Z_p has roughly p points on it.

2.2.1 Point addition on an elliptic curve over a finite field

If $P=(x_P,y_P)$ and $Q=(x_Q,y_Q)$ be two points in $E_p(a,b)$, then $R=(x_R,y_R)=P+Q$ is calculated by the following rule [111]:

$$x_R = (\mu^2 - x_P - x_Q) \pmod p$$
$$y_R = (\mu(x_P - x_R) - y_P) \pmod p$$

where $\mu = \begin{cases} \frac{y_Q-y_P}{x_Q-x_P} \pmod p, \text{if } P \neq -Q \\ \frac{3{x_P}^2+a}{2y_P} \pmod p, \text{if } P = Q. \end{cases}$

The case $P=Q$ is often referred as doubling the point, and it is represented as $2.P$.

2.2.2 Scalar multiplication on an elliptic curve over a finite field

The scalar multiplication of an elliptic curve point $P \in E_p(a,b)$ is denoted by $k.P$, where $k \in Z_p^* = \{1,2,\ldots,p-1\}$ is a scalar, and it is achieved by using repeated point additions and doubling the point operations. For example, if $P \in E_p(a,b)$, then $23.P$ is calculated as $23.P = P+P+\ldots+P$ (23 times). More efficient approach allows to compute $23.P = 2.(2.(2.(2.P)+P)+P)+P$ using three point additions and four doubling the point operations.

2.2.3 Elliptic curve discrete logarithm problem (ECDLP)

Given two points $P,Q \in E_p(a,b)$ where $Q=k.P$ and $k \in Z_p^*$ is a scalar. Computing k from P and Q is computationally infeasible if p is sufficiently large (for example, p may be 160 bits prime). This problem is referred to as "elliptic curve discrete logarithm problem (ECDLP)". Formally, ECDLP is defined as follows [56].

Definition 2.2 (Elliptic curve discrete logarithm problem (ECDLP))**.** *Let $E_p(a,b)$ be an elliptic curve over a prime field $GF(p)$, and $P,Q \in E_p(a,b)$ where $Q=k.P$ and $k \in Z_p^*$.*

Instance: (P,Q,r) for $k,r \in Z_p^*$.

*Output: **Yes**, if $Q=r.P$, i.e., $k=r$; **No**, otherwise.*

Consider the following distributions:

$\Delta_{real} = \{k \in Z_p^*, A = P, B = Q(= k.P), C = k : (A, B, C)\}$,
$\Delta_{rand} = \{k, r \in Z_p^*, A = P, B = Q(= k.P), C = r : (A, B, C)\}$.

The advantage of any probabilistic, polynomial-time, 0/1-valued (false/true-valued) distinguisher $\mathcal{D}$ in solving ECDLP on $E_p(a, b)$ is defined as follows:

$$\begin{aligned} Adv^{ECDLP}_{\mathcal{D},E_p(a,b)} &= |Pr[(A, B, C) \leftarrow \Delta_{real} : \mathcal{D}(A, B, C) = 1] \\ &\quad -Pr[(A, B, C) \leftarrow \Delta_{rand} : \mathcal{D}(A, B, C) = 1]|, \end{aligned}$$

where the probablity $Pr[\cdot]$ is taken over random choices of k and r. $\mathcal{D}$ is said to be an (t, ϵ)-ECDLP distinguisher for $E_p(a, b)$ if $\mathcal{D}$ runs at most in time t such that $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)} \geq \epsilon$.

ECDLP assumption: There exists no (t, ϵ)-ECDLP distinguisher for $E_p(a, b)$. In other words, for every probabilistic, polynomial-time 0/1-valued distinguisher $\mathcal{D}$, we have, $Adv^{ECDLP}_{\mathcal{D},E_p(a,b)} \leq \epsilon$.

The security of the ECC depends on the intractability of ECDLP. There are several algorithms, such as Pollard's rho method [159] and baby-step giant-step method [180] to solve ECDLP in exponential or sub-exponential time complexity. However, no efficient polynomial-time algorithm exists for solving ECDLP so far in the literature, and solving ECDLP is an open problem.

2.2.4 Elliptic curve computational Diffie-Hellman problem (ECCDHP)

The elliptic curve Diffie-Hellman is a variant of the Diffie-Hellman key agreement protocol between two communicating parties over a public channel that utilizes elliptic curve private and public keys pair to generate a shared secret key. The generated shared secret key is used for subsequent secure communication between two parties. The "elliptic curve computational Diffie-Hellman problem (ECCDHP)" is defined as follows.

Definition 2.3 (Elliptic curve computational Diffie-Hellman problem (ECCDHP))**.** *Let P be a point in $E_p(a, b)$. The ECCDHP states that given the points $k_1.P \in E_p(a, b)$ and $k_2.P \in E_p(a, b)$ where $k_1, k_2 \in Z_p^*$, it is computationally infeasible to compute $k_1.k_2.P$.*

2.2.5 Elliptic curve decisional Diffie-Hellman problem (ECDDHP)

The "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" is a computationally hard problem, which forms the basis for many security protocols designed using elliptic curve cryptography (ECC). The ECDDHP is defined as follows.

Definition 2.4 (Elliptic curve decisional Diffie-Hellman problem (ECDDHP))**.** *Let P be a point in $E_p(a, b)$. The ECDDHP states that given a quadruple $(P, k_1.P, k_2.P, k_3.P)$, it is impossible to distinguish between whether $k_3 = k_1.k_2$ or a uniform value, where $k_1, k_2, k_3 \in Z_p^*$.*

The ECDLP, ECCDHP and ECDDHP are computationally infeasible when p is sufficiently large. It is worth noticing that p having more than 160 bits is sufficient to make ECDLP, ECCDHP and ECDDHP intractable.

2.2.6 Elliptic curve digital signature

Digital signatures are used to authenticate messages or digital documents while ensuring non-repudiation and integrity. The signature algorithms employ asymmetric or public key cryptography techniques and consist of three phases: 1) key generation, 2) signature generation and 3) signature verification. Elliptic curve digital signature algorithm (ECDSA) is one such variant of the original digital signature algorithm and it's phases have been explained below.

- **Key generation:** First, the system is setup by choosing an elliptic curve $E_p(a, b)$ and its base point G. Then, every entity chooses its private key $d \in Z_p^*$ and computes its corresponding public key as $e = d.G$.

- **Signature generation:** Consider an entity with parameters $E_p(a, b), h(\cdot), e, G, p$ where $h(\cdot)$ is a collision-resistant hash function. Suppose m is the message to be signed. Using its key pair (d, e) and a chosen random number $k \in Z_p^*$, the entity computes the signature as follows:

$$\begin{aligned} k.G &= (x_1, y_1), \\ c &= h(m), \\ r &= x_1 \pmod p, \\ s &= l^{-1}(c + d.r) \pmod p. \end{aligned}$$

  If either $r = 0$ or $s = 0$, the algorithm restarts. Otherwise, (r,s) is the signature of the sender for message m. The signer then sends the signed message $\langle m, (r, s)\rangle$ to the verifier.

- **Signature verification:** The verifier verifies the signature (r, s) by first checking if

$r, s \in Z_p^*$, and then by using the signer's public parameters computes the following:

$$\begin{aligned} c &= h(m), \\ u &= s^{-1} \pmod{p}, \\ v &= c.u \pmod{p}, \\ w &= r.u \pmod{p}, \\ t &= v.G + w.e \\ &= (t_x, t_y), \\ r^* &= t_x \pmod{p}. \end{aligned}$$

The signature is accepted by the verifier only if $r^* = r$.

2.2.7 ECC *versus* RSA

System security level is an ever present concern. The recommended security for current systems is 128-bits in terms of both the key length and the algorithm implementation. For example, while RSA public-key cryptosystem is based on the computational complexity of factorization, ECC utilizes the discrete logarithm problem. The implementation impacts system parameters like processing capabilities and energy consumption. Jansma and Arrendondo did a comparative performance analysis of the ECC and RSA signature algorithms on an Intel P4 2.0 GHz machine with 512MB of RAM. Their results as recorded in [101] indicate that ECC provides equivalent security as RSA for smaller key sizes. Also, the computation time for key and signature generation is significantly less in ECC than in RSA, especially for larger key sizes as ECC does not require large prime number generation like RSA. Table 2.3 presents a comparison of key size and computation time for signature generation for each key length. The significantly low computation time and resources required in ECC make it suitable for resource constrained environments.

2.3 Simulation tools for verifying Internet security protocols

The formal security verification using automated software tools has gained a huge popularity among the researchers in the security domain. The security protocols can be verified by several verification tools to assure the security protocols that they are secure against some

Table 2.3: Comparison of key length and computation time for signature generation [101]

Key length (in bits)		Signature generation time (in seconds)	
ECC	RSA	ECC	RSA
163	1024	0.08	0.16
233	2240	0.18	7.47
283	3072	0.27	9.80
409	7680	0.64	133.90
571	15360	1.44	679.06

attacks, such as replay attack and man-in-the-middle attack. There are several formal security verification tools under the Dolev-Yao (DY) model [67] for security protocols (for instance, access control, authentication, and key agreement), such as: a) "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [17], b) ProVerif [32], and c) Scyther [50].

2.3.1 AVISPA tool

In this chapter, we provide details description of AVISPA, which is a broadly-accepted automated software tool for the formal security verification for the presented security protocols in chapters 4, 5 and 6.

AVISPA [17] is treated as a push-button approach for the validating the "Internet security-sensitive protocols" as well as its applications. It supports a modular as well as expressive formal language for identifying the security protocols together with their security possessions. It also unites various back-ends that execute a heterogeneity of state-of-the-art automatic analysis methods. There are four back-ends which are implemented in AVISPA and these are: 1) "On-the-fly mode-checker (OFMC)", 2) "Constraint-logic-based Attack Searcher (CL-AtSe)", 3) "SAT-based Model Checker (SATMC)", and 4) "Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)", which are shown in Figure 2.1. The details of these back-ends can be found in [17].

The tested protocols need to be implemented using the "High-Level Protocol Specification Language (HLPSL)" with *hlpsl* file extension. The code written in HLPSL is then converted into the "Intermediate Format (IF)" using the HLPSL2IF translator, which is fed into one of the available four backends to get the "Output Format (OF)". It is a role based language

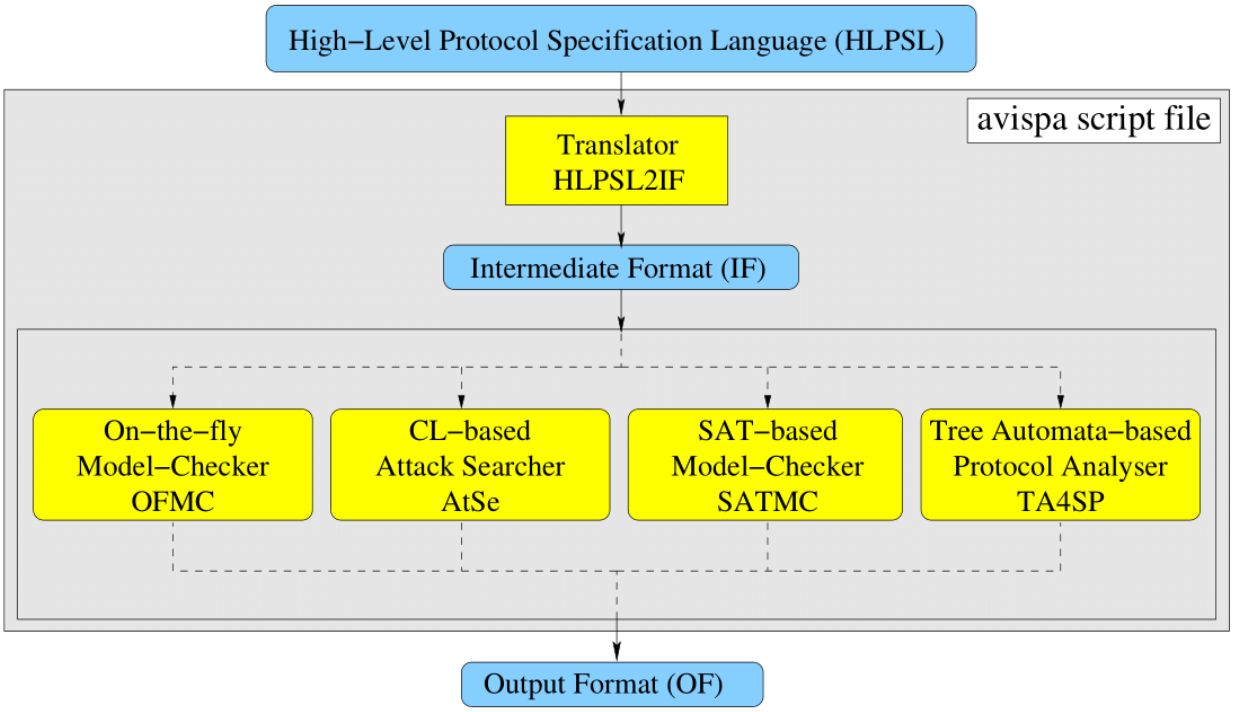

Figure 2.1: Architecture of AVISPA tool v.1.1 (adopted from [17])

where the roles can be: a) basic role, b) composition role, c) session role, and d) environment role and goal. Each of these role is independent from the other roles, acquires some initial data by parameters or transmits with the other roles by communication channels. The basic role represents the role of each participant, whereas the composition role speaks for the structure of the basic role. The session role indicates a single session for the protocol and it is parameterized by all the necessary variables those are required for one session. The final role for the HLPSL is called environment, which holds the session composition as well as the constants which are globally defined. The goal has a duty to check the message authenticity as well as the confidentiality (privacy) of secret messages/keys.

A role is defined along with the following characteristics:

- ***Local declaration:*** The variables and their types are declared.
- ***Constants declaration:*** Under this declaration, the constant variables and their types can be declared, which are not local to the role and can be utilized in another roles.
- ***Initialization:*** Here, the local variables are initialized.

- ***Accept declaration:*** The role can considered as done in this condition.
- ***Intruder knowledge declaration:*** At the starting of the role implementation, a set of variables are provided to the intruder. The intruder is always denoted by the special indentifier "*i*".

Each of the following four goals holds the following credentials:

- *secret(A,id,B):* It means the secret A is shared with the agents in a set B characterized by the protocol identifier, id.
- *witness(A,B,id,C):* It represents a "weak authentication" of A by B on C, which is characterized by the protocol identifier, id.
- *request(B,A,id,C):* It is for a "strong authentication" of A by B on E, which is characterized by the protocol identifier, id.
- *wrequest(B,A,id,E):* It is identical with the request(), but it is applied for a "weak authentication".

The HLPSL specification of a tested security protocol is first translated to its "Intermediate Format (IF)" using a translator available in AVISPA, known as the HLPSL2IF translator. The IF is taken as input to one of the four available back-ends (OFMC, CL-AtSe, SATMC and TA4SP) to convert to the "Output Format (OF)". The OF consists of the following important sections [203]:

- SUMMARY: It states "whether the tested protocol is safe, unsafe, or whether the analysis is inconclusive".
- DETAILS: It tells "a detailed explanation of why the tested protocol is concluded as safe, or under what conditions the test application or protocol is exploitable using an attack, or why the analysis is inconclusive".
- PROTOCOL: It defines the "HLPSL specification of the target protocol in IF".
- GOAL: It specifies "the goal of the analysis which is being performed by AVISPA using HLPSL specification".
- BACKEND: It provides "the name of the back-end that is used for the analysis, that is, one of OFMC, CL-AtSe, SATMC and TA4SP".

- Finally, we have "the trace of a possible vulnerability to the target protocol, if any, along with some useful statistics and relevant comments".

We require three verifications that are needed to be checked for a tested security protocol: a) "executability checking on non-trivial HLPSL specifications", b) "replay attack checking", and c) "Dolev-Yao (DY) threat model checking" [67]. The executability check is extremely needed to assure that the tested protocol can reach to a state where a possible attack can occur during the protocol execution. To check the "replay attack on a tested protocol", the backends check whether the legitimate agents can execute "the specified protocol by performing a search of a passive intruder". In addition, the backends also verify whether any man-in-the-middle attack can be performed by the intruder (i) for the DY threat model checking.

2.3.2 Other automated validation tools

ProVerif [32] is an "automatic symbolic protocol verifier that supports a wide range of cryptographic primitives, defined by rewrite rules or equations". Various security attributes, such as authentication, secrecy and process equivalences can be proved through this tool for "an unbounded message space and an unbounded number of sessions".

Scyther [50] supports a "graphical user interface (GUI)" which accompaniments the "command-line" and "Python scripting interfaces". This tool containing the "command-line" and "scripting interfaces" makes its utilization for "large-scale protocol verification tests". The main feature provided in Scyther is that it assures termination whilst permitting to prove "correctness of protocols for an unbounded number of sessions", and it alternatively produce the output as the proof tree with the help of its backend.

2.4 Blockchain technology and its consensus protocols

In this section, we provide a brief overview of the blockchain technology including its various applications, and the consensus algorithms used for mining the blocks in a Peer-to-Peer (P2P) blockchain network.

2.4.1 Overview of blockchain

In 1991, Haber and Stornetta [88] introduced an abstraction for securing a chain of time-stamped digital documents or records. However, in 2018, Satoshi Nakamoto published a

formal draft known as "Bitcoin: A peer-to-peer electronic cash system" [148] that provided the implementation of the blockchain technology using time-stamping concept in a cryptocurrency scenario, popularly known as Bitcoin [4, 142].

A blockchain is considered as as shared distributed database or ledger containing a series of blocks or chains that facilitates storing the information in the form of transactions. Blockchain is formed by a Peer-to-Peer (P2P) distributed network, where the nodes are distributed. Thus, the blockchain is decentralized in nature. The information stored in the blockchain is protected and it is secured by the cryptographic techniques, like "elliptic curve cryptography (ECC)" encryption/decryption, ECC-based digital signature, and one-way cryptographic hashing. The initial block in the blockchain is referred as the "Genesis" block and the subsequent blocks are linked with the previous blocks containing the cryptographic hash values of the previous blocks. A typical blockchain structure with three nodes is shown in Figure 2.2. A block in the blockchain has two parts: 1) header and 2) payload [245]. The block header contains various fields, such as previous block hash, block version, timestamp, Merkle tree root value, etc., whereas the block payload contains transactions. Finally, the current block hash and signature on the block can be put in the block as well.

There are several advantages of using the blockchain technology as a service, because "decentralization", "immutability", "transparency", "confidentiality" and "trust" are maintained [146]. Transparency property means that "when an entity's real identity is made secure, one can still view all the transactions that were done by their public addresses". Immutability property allows that "once a block having the information is inserted into the private/public blockchain, it can not be tampered later". Blockchain is of three categories [4]: 1) "public blockchain", 2) "private blockchain" and 3) "consortium blockchain". The public blockckchain is "open to anybody to join, access, send, verify and receive transactions of the blocks in the network". Some applications that use public blockchain are "cryptocurrency (Bitcoin)" [148] and "Ethereum". In a private blockckchain, it is a considered as a fully trusted network. In this case, "the access is only granted to a particular entity or a group of trusted entities" and also "the owner of the network mainly decides which entity will perform a specific task". A consortium blockchain is treated as a "combination of both public and private blockchains". Consensus algorithms are needed in order to achieve "consensus among the nodes involved in a peer-to-peer (P2P) blockchain network". Some broadly-accepted consensus algorithms are "Byzantine Fault Tolerance (BFT)", "Practical Byzantine Fault Tolerance (PBFT)", "Proof-of-Work (PoW)", and "Proof-of-Stake (PoS)" [11].

- **Public (Permissionless) blockchain**: In this case, everyone has the right to "join,

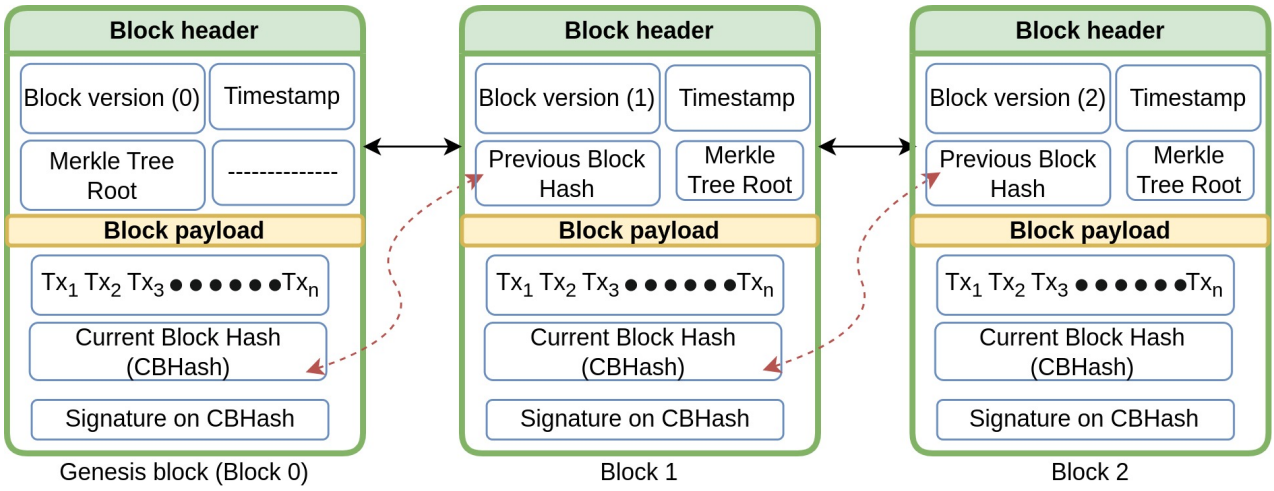

Figure 2.2: Structure of blockchain (adopted from [172])

access, send, verify and receive transactions of the blocks" in the blockchain to create a consensus. Bitcoin and litecoin are the popular cryptocurrencies that use the public blockchain [4].

- **Private blockchain:** Private (permissioned) blockchains are created and also maintained by the private institutions (organizations). In such case, the creators will have the access control over mining process as well as the consensus algorithm that are followed by the private blockchain network. In other words, such type of blockchain is considered as the distributed network with private control itself. One of the popular use of the private blockchain is the healthcare system, where the data pertaining to the patients, doctors, nurses, staffs, pharmacy, etc. are strictly confidential and private. Hyperledger fabric or Sawtooth, and Ripple allow to form a private blockchain [4].

- **Consortium blockchain:** Under a consortium blockchain, a set of multiple financial organizations (institutions) exists, where each financial organization will have its private blockchain. In this type of blockchain, only a pre-selected set of peer nodes in the P2P blockchain network are permitted to control the consensus process [4, 220]. Some open-source consortium solutions include Hyperledger, Corda Ripple, Quorum, Ethermint, and Multichain. An example of a consortium blockchain is as follows. Consider a "supply-chain consortium blockchain between a logistics company and a manufacturer", where solving the problems in either of organizations would be beneficial for improving the outcomes for both [7].

2.4.2 Consensus algorithms

A consensus algorithm is defined as a process where the majority of the peer nodes of the blockchain network need to reach to a common agreement for adding a block into the blockchain (distributed ledger). In the following, we discuss some widely-used consensus algorithms.

- ***Proof of Work (PoW)***: The abstraction of PoW was narrated from "reusable proof of work" concept proposed by Finney in 2004 using Secure Hash Algorithm (SHA-1) (which produces 160-bit hash output) [78]. It is the most popular consensus algorithm used in the Bitcoin. PoW is a "compute-intensive-based consensus algorithm" [98] that was implemented in 2008 by Nakamoto [148]. In PoW, if a new block needs to be added into the blockchain network, the miner node needs to solve mathematical or cryptographic puzzles, which is also referred as "proof of work problem". The new block will only get validated, if the sufficient number of P2P nodes agree. However, this process is very expensive to add a block, becausethe miner requires a huge resource to solve the puzzles. Additionally, selfish mining and "distributed denial of service (DDoS)" attacks are the most common attacks that occurred in PoW.

- ***Proof of Stake (PoS)***: The PoS is also referred to as "capability based consensus protocol". Here, the miner is treated as a validator. To become a validator in PoS, the members need to keep coins or tokens as a stake. However, the validators get the transaction fees as a reward. A block in the network is first selected randomly which has a large currency in stake. As compared to PoW, PoS requires relatively less computational resources. In other words, PoS is considered as a relatively safe protocol and inferior for an attack in the network.

- ***Delegated Proof of Stake (DPoS)***: Larimer in the year 2013 introduced the concept of DPoS from the project, BitShares [1]. In DPoS, the blocks are validated by a set of nodes (referred to as delegates or witnesses) in the network and each stakeholder can give a vote in the election process [69]. Since a group of delegates controls a decision, it is considered as a partially centralized consensus protocol.

- ***Byzantine Fault Tolerance (BFT)***: The functioning of BFT algorithm is based on the voting mechanism which is used to create and add a block into the blockchain. It gives conclusiveness for adding a block, which eliminates the possibility of rollbacks in PoW. However, BFT (also known as "compute-intensive-based protocol") requires

less energy utilization. It has the ability to handle the Byzantine faults [114] in the distrustful P2P environment, where a node can behave deceptively or maliciously. However, the scalability of BFT is not considered as efficient. Moreover, there are several algorithms designed based on BFT, which include i) "Practical Byzantine Fault Tolerance (PBFT)", ii) "Delegated Byzantine Fault Tolerance (DBFT)", and iii) "Federated Byzantine Agreement (FBA)" [201, 245].

- ***Practical Byzantine Fault Tolerance (PBFT)***: It is considered as a voting-based consensus algorithm that was introduced by Castro and Liskov [38] in the year 1999. If there exist disloyal or faithless nodes in the asynchronous distributed P2P network, PBFT can assure the consensus process. It can handle the consensus even if the number of faulty nodes in the network satisfies $n_f < \frac{z-1}{3}$, that is, $z > 3n_f + 1$, where z is the total participated nodes in the P2P network.

- ***Ripple Protocol Consensus Algorithm (RPCA):*** In order to maintain the correctness as well as agreement of the network, RPCA is applied every few seconds by all the nodes in the P2P network. The present ledger is treated as "closed" whenever the consensus is reached, and it then comes the "last-closed ledger". If the consensus process becomes successful, and there is no fork in the network as well, the "last-closed ledger" that is maintained by all the nodes in the P2P network will be considered as identical [176].

  In RPCA, a transaction only gets approved when 80% of the "Unique Node List (UNL)" of a server agrees with it. Therefore, as long as 80% of the UNL is "honest", RPCA does not allow to approve any fraudulent transactions. In RPCA, for a UNL of n peer nodes in the P2P network, the consensus maintains correctness if the condition: $f \leq \frac{n-1}{5}$ is satisfied, where f is the number of Byzantine failures.

2.4.3 Some blockchain applications

1) Blockchain application in smart grid

A blockchain-based smart grid system contains several entities, such as trusted registration authority (RA), service providers (SPs), IoT-enabled smart meters (SMs), and users associated with a smart meter. Figure 2.3 shows the role of P2P network of service providers in a smart grid. SPs organize the electricity allocation and energy trading system, and SMs are responsible for monitoring the power utilization and they maintain the pricing to the consumers

(users). SMs can be deployed in the homes, and an attacker may capture some SMs and use the secure data stored into it [105]. The communications between SPs and SMs must be secure so that passive/active attacks should be not possible [199]. To ensure the security and privacy of the users' private information, it is extremely required to design a secure and efficient access control scheme between SMs and SPs. With the help from the blockchain technology, the secure data can be stored in the form of blocks in a private blockchain. The SPs involved in the P2P SP network are responsible in validating the new blocks before adding them into the blockchain using the consensus algorithm. To mitigate these issue, we aim to design a blockchain-based access control mechanism in an IoT-enabled smart grid environment.

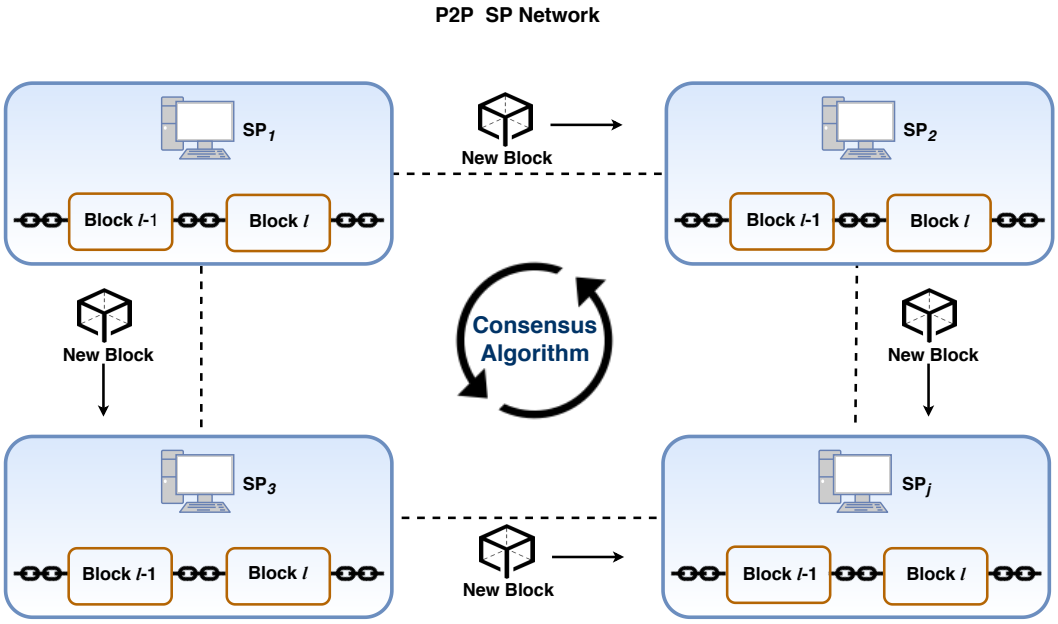

Figure 2.3: Blockchain-based P2P network of service providers (SP_j) in smart grid (adapted from [239])

2) Blockchain application in healthcare

A healthcare application using the blockchain technology is shown in Figure 2.4. Such an application uses private blockchain in a group of trusted hospitals. In the network model shown in Figure 2.5, it is assumed that the trusted Hospital Authority (HA_i) of i^{th} hospital $Hospital_i$ will receive the private and confidential data securely with the help of access control mechanism from its respective authorized registered users [173]. The HA_i being the *Miner node* constructs the blocks in the blockchain. All the transactions residing in the blocks are

encrypted using the secret key of the HA_i which is shared with its own users as well as other trusted HA_k of other hospitals $Hospital_k$. After validating the transactions in a block, an intended verifier (for example, a doctor in hospital) can see the confidential and private data of the $Hospital_i$ for diagnostic purpose.

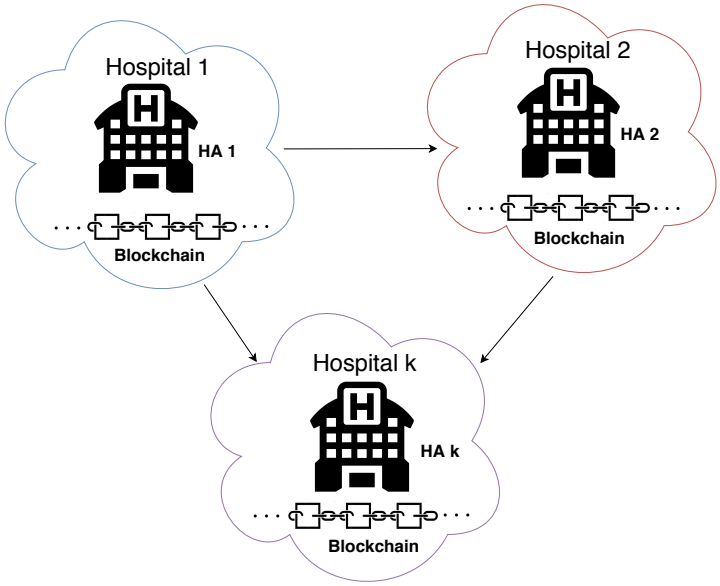

Figure 2.4: Healthcare system with blockchain technology

3) Blockchain application in cyber-physical systems

Modern Industrial Cyber-Physical Systems (ICPS) utilizes the advanced "Artificial Intelligence (AI)" and "Machine Learning (ML)" approaches in order to enhance several issues, such as "scalability", "speed" along with accuracy of ICPS security. In this paper, we mainly focus on "accuracy of ICPS security" because of possibility of "data poisoning attacks". To alleviate such concerns, we design a new blockchain-enabled signature-based key management protocol in an ICPS environment, which will allow the IoT smart devices to securely communicate with their respective gateway nodes. The gateway nodes after forming transactions containing secure data from the smart devices forward those transactions securely to their attached fog servers. Later, the cloud servers in the blockchain center are in charge of creating, validating and appending the blocks in the private blockchain [54].

There are various advantages for using distributed storage against using a centralized system in ICPS. The distributed storage in ICPS can make a system more robust and protect

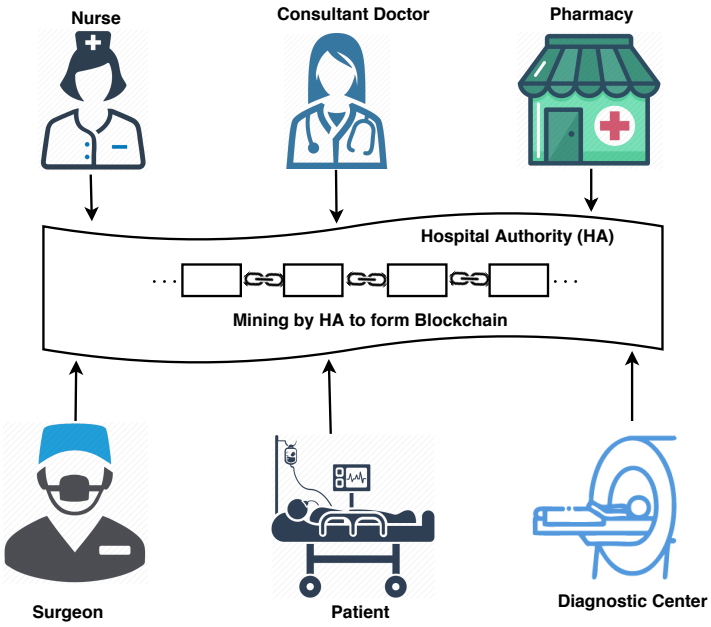

Figure 2.5: Network model with blockchain technology

the system from a single point of failure, low latency and cost effectiveness [116]. In particular, a blockchain-based ICPS system can provide transparency, immutability, and strong security against various attacks. Moreover, the AI/ML based algorithms are applied on the valid data pertaining in the blocks of the blockchain for correct predictions that will further be very useful for Big data analytics.

A blockchain-enabled ICPS network model is presented in Figure 2.6. Based on each ICPS application belonging to a company, say Company A, we partition the deployed/installed IoT-enabled smart devices into a number of disjoint groups (clusters), say Gr_j $(j = 1, 2, \cdots, n_{gr})$. Each Gr_j will contain a number of IoT smart devices, say SD_i $(i = 1, 2, \cdots, n_{sd})$ and a gateway node GWN_j. All the IoT smart devices SD_i in Gr_j will communicate with their GWN_j securely using the created secret keys through the key management procedure. Each GWN_j in group Gr_j is attached with a fog server, say FS_k $(k = 1, 2, \cdots, n_{fs})$.

The data securely brought by the gateway node GWN_j from its IoT (smart) devices SD_i is used to form the transactions and forward them securely to its FS_k. FS_k will be responsible for creating a partial block from the gathered secure transactions and forwarding the partial blocks to a cloud server CS_l $(l = 1, 2, \cdots, n_{cs})$ for verifying and adding the complete blocks using some consensus algorithm, say the "Practical Byzantine Fault Tolerance (PBFT) consensus algorithm" [38] under the P2P cloud servers network. The cloud servers belong to

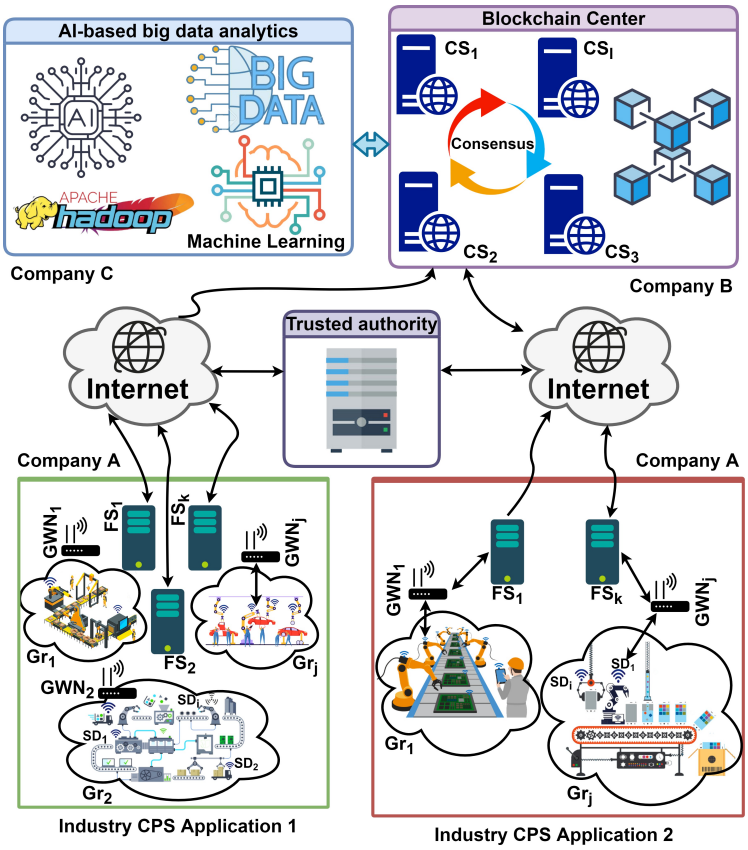

Figure 2.6: Blockchain-enabled ICPS (adopted from [54]

another company, say Company B. In addition, we have "AI-based Big data analytic center" belonging to another company, say Company C, where the genuine transactional data stored in private blockchain in the Blockchain Center (BC) are used for appropriate predictions with AI/MI algorithms.

4) Other blockchain applications

There are also several other blockchain applications, like smart manufacturing, Internet of Vehicles (IoV) [43, 70, 71, 72, 77, 163, 165, 186, 191, 213, 229], Internet of Drones (IoD) [21, 24, 27, 28, 30, 143, 212, 231], supply chains [100, 147, 179, 223, 238], food industry [79, 200, 208, 242], smart manufacturing [117, 118, 119, 196, 243], smart grids [29, 90, 130, 145, 157, 210, 232], and healthcare [16, 164, 166, 185, 197, 241]. Finally, Blockchain of Things (BCoT) is illustrated in Figure 2.7.

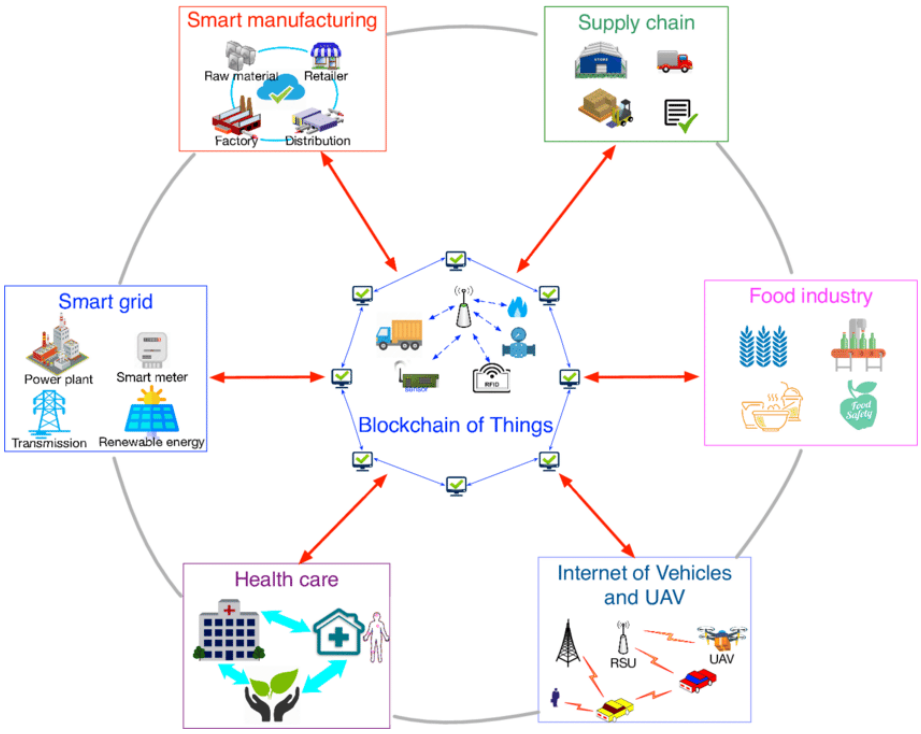

Figure 2.7: Blockchain of Things (BCoT)

2.5 Merkle tree and its use in blockchain

A Merkle tree is a data structure that is applied in many computer science applications. It is also known as "binary hash trees". Merkle trees are generated by repeatedly comtuing the hashing pairs of the nodes (here transactions) until there is only one hash remained. This hash is known as the "Merkle root" or the "root hash". The construction of the Merkle trees takes place in a bottom-up method as shown in Figure 2.8, where there are eight transactions Tx_i, $i = 1, 2, \ldots, 8$. Here, H_i: Hash of i^{th} transaction Tx_i ($H_{Tx_i} = H(Tx_i)$); $H_{Tx_1Tx_2} = H_{Tx_1} \oplus H_{Tx_2}$ or $H(H_{Tx_1}|H_{Tx_2})$; and $H_{12345678}$: Merkle root.

Now, we discuss the use of the Merkle tree in blockchain [81] as shown in Figure 2.9. In this figure, we assume for the sake of simplicity that there are four transactions and using these transactions the Merkle tree root is calculated. After that the Merkle root is stored in the header of the block 1, where block 0 is the Genesis block in the blockchain. Note that the pointer of the Merkle tree root is also stored in the block so that the entire tree can be fetched during the verification process of the transactions containing in that block.

In Figure 2.10, we have shown the mechanism for verifying a transaction, say Y. In order to do so, we only require to know $H(WX)$, $H(Y)$, $H(Z)$ and $H(WXYZ)$. Once $H(WXYZ)$

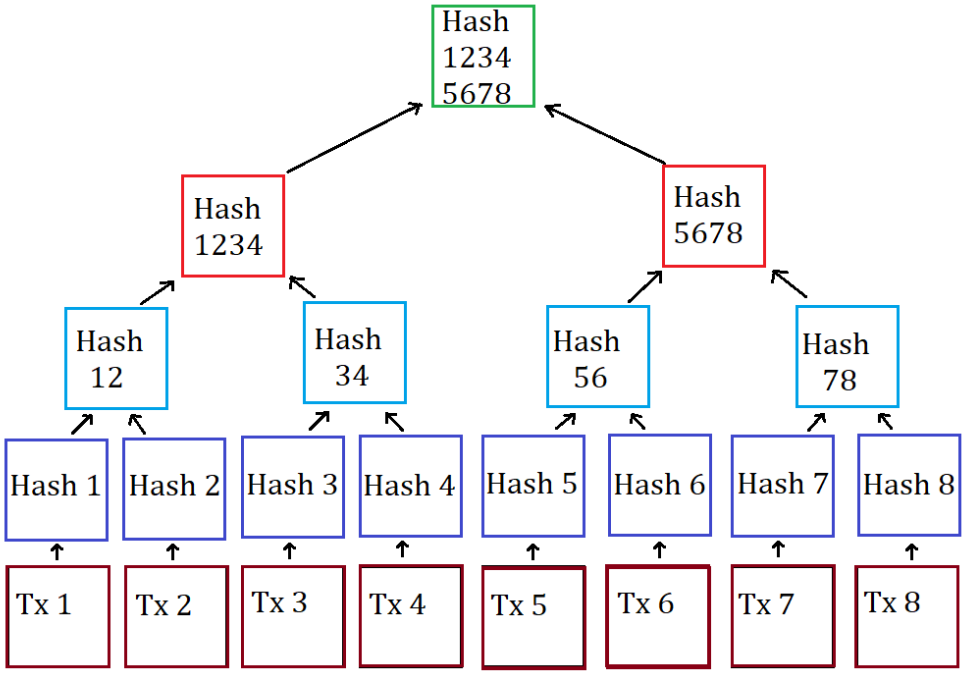

Figure 2.8: Creation of Merkle tree

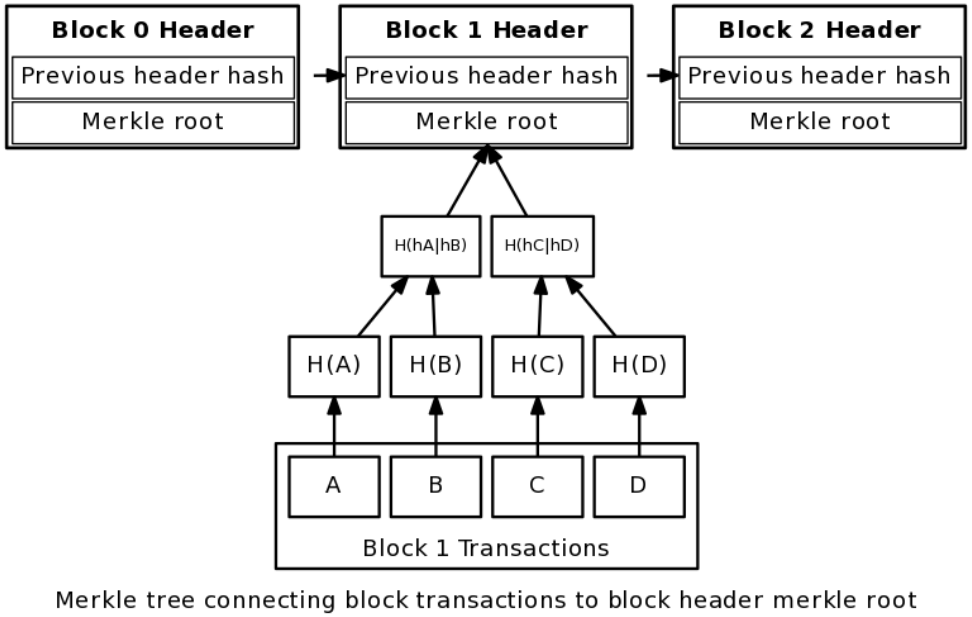

Figure 2.9: Use of Merkle tree in blockchain [81]

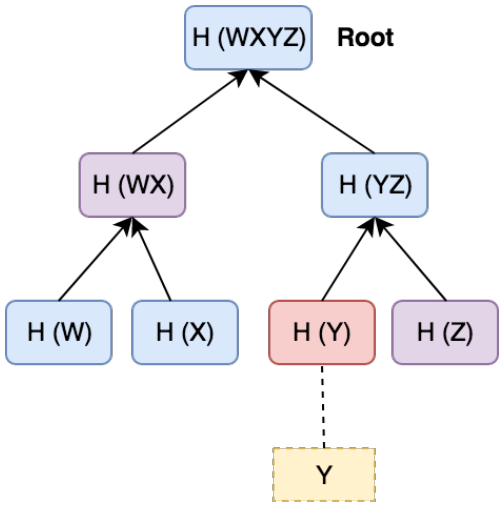

Figure 2.10: Verifying transactions using Merkle root

is re-calculated, it is checked against the value stored in the block's header. If there is no mis-match, the transaction Y is treated as authentic one.

In the context of the blockchain, a Merkle tree can store all the transactions residing in a block by means of producing a "digital fingerprint of the entire set of transactions". Thus, the Markle tree allows a user to check if "a transaction can be included in a block or not". Finally, we list that Merkle trees will have three major benefits:

- It provides the integrity and validity of data (transactions) in a block.
- It helps in saving the "memory or disk space as the proofs, computationally easy and fast".
- The proofs and management need small amounts of information that will be transmitted across the P2P network.

2.6 Summary

In this chapter, we have discussed those mathematical preliminaries that are useful in discussing and analyzing the proposed schemes in the subsequent contributory chapters in this thesis work. We have discussed the one-way cryptographic hash functions and their important properties. Next, we have discussed "elliptic curve cryptography (ECC)", its properties and also its related computationally hard problems. We have then discussed about the automated software validation tools for formal security verification for the Internet security protocols to

verify whether a security protocol is safe or unsafe. We have discussed the blockchain technology, its various applications and underlying consensus protocols. Finally, we have discussed the importance of the Merkle tree in the context of the transaction verification in a block into the blockchain.

Chapter 3

Literature Survey

This chapter presents many existing security protocols in Internet of Things (IoT) environment as well as wireless sensor networks (WSNs), which are based on access control, authentication, and key agreement.

3.1 Bakground

3.1.1 Security requirements in IoT environment

Khalil *et al.* [106] discussed the integration of "Wireless Sensor Networks (WSNs) into IoT". While the sensor nodes in WSNs are resource limited in nature, the majority of IoT smart devices are also resource constrained. Therefore, the security protocols proposed in WSNs can be also applied to the IoT applications [60].

- *Authentication:* It involves authentication of sensing devices, users and gateway nodes before allowing access to a restricted resource, or revealing crucial information.
- *Integrity:* The message or the entity under consideration must not be changed to ensure integrity.
- *Confidentiality:* Confidentiality or privacy of the wireless communication channel protects from the unauthorized disclosure of information.
- *Availability:* The relevant network services should be made available to authorized users even under denial-of-service attacks on the system.
- *Non-repudiation:* It requires to prevent a mischievous entity from hiding his/her actions.

- *Authorization:* It guarantees that only the legitimate IoT sensing (smart) devices can supply information to network services.
- *Freshness:* The information needs to be fresh and the old messages cannot be replayed by any adversary.

Apart from the above security requirements, the following two important security properties should also be satisfied:

- *Forward secrecy:* If "an IoT sensing node quits the network", any future messages after its exit must be prohibited.
- *Backward secrecy:* If "a new IoT sensing node is added in the network," it must not read any previously transmitted message.

3.1.2 Security attacks in IoT environment

Based on the aforementioned security requirements discussed in Section 3.1.1, several attacks should be prevented in the design of security protocols for IoT environment. Some of these attacks include the following [60]:

- *Replay attack:* A replay attack is one in which "an adversary, $\mathcal{A}$ attempts to mislead another authorized entity by reusing the information during the transmission".
- *Man-in-the-middle attack:* In this attack, $\mathcal{A}$ intercepts the "transmitted messages" and tries to "change/delete/modify the contents of the messages delivered to the recipients" on the fly.
- *Stolen-verifier attack:* This attack can occur if the GWN in the IoT network stores any verifier/password table for user/device verification. It is important that the design of security protocols in IoT should not store any verifier/password table for verification in order for the protocols need to be resilient against this type of attack.
- *Password guessing attack:* In a password-based scheme, $\mathcal{A}$ may attempt to guess the password of a legal registered user either online or offline mode with the help of the eavesdropped messages and also stored credentials in the system or a user's smart card (mobile device). Thus, the credentials should be stored in such a way that even if the user's smart card (mobile device) is lost/stolen, $\mathcal{A}$ should not be able to derive the user's secret credentials.

- *Password change attack:* In this attack, $\mathcal{A}$ may try to change the password of an authorized registered user. For instance, if a smart card-based scheme of the smart card (mobile device) of a legitimate user is compromised, $\mathcal{A}$ can breach the stored information in that smart card to replace the user's password with a fake password.

- *Denial-of-Service attack:* A Denial-of-Service (DoS) attack is any event that prevents a system's or network' capability to perform its expected function. A DoS attack can happen due to several factors including software bugs, hardware failures, environmental conditions or resource depletion [221].

- *Privileged-insider attack:* In this type of attack, a trusted user within the organization (also known as an insider) can act as a privileged-insider attacker. An insider user can obtain the secret credentials of a registered user during the registration process of an authentication scheme, and then try to misuse those credentials.

- *Impersonation attack:* In an impersonation attack, an attacker may attempt to falsify a fake message to defraud other recipient entities in a network on behalf of a sending entity. In such an attack, the receiver will believe that the message has been received from a legitimate entity.

- *Resilience against sensing device capture attack:* In IoT environment, except the GWN the IoT sensing devices are not physically protected. Hence, there is a possibility of physical capturing of the sensing devices by an attacker $\mathcal{A}$. Next, $\mathcal{A}$ can use the extracted information stored in those captured sensing devices to compromise communication between other non-compromised sensing devices.

- *Resilience against new sensing devices deployment attacks:* An access control scheme designed in the IoT environment must be resilient against several attacks, such as "illegal sensing devices deployment", "Sybil", "sensing device replication" and "wormhole" attacks. In a large network, an Intrusion Detection System (IDS) helps to detect intruders and prevent attacks in the IoT system. IDS mostly works to prevent against internal attacks and it can be categorized as flow based, payload based, and hybrid based IDS [167]. In flow based IDS, it examine the frequency and the interval of the transmitted message then based on its unusual behavior IDS provide warning. Moreover, payload based IDS investigate based on the payload of a message. In addition, the hybrid based IDS consist of both payload and flow based IDS. However, recent days various IDS also uses machine learning techniques to classify the attacks [51].

– In a wormhole attack [93], $\mathcal{A}$ tries to tunnel the information between two distant locations with the help of an in-band or out-of-band channel. A wormhole can deceive and bypass a large amount of network traffic. Thus, it can enable the wormhole devices to easily obtain and control the network traffic.

– A Sybil attack occurs when a malicious sensing device falsely assumes multiple identities [68, 150]. The claimed identities may or may not be from the existing sensing devices' identities.

– In a sensing device replication attack [158], $\mathcal{A}$ intentionally creates various replicas of a compromised sensing device which are then inserted into the network.

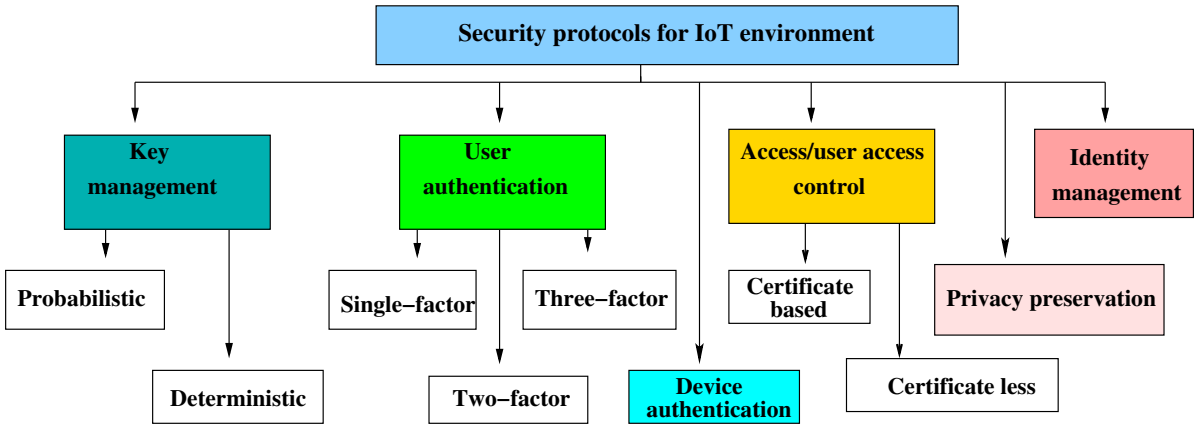

Figure 3.1: A taxonomy of security protocols in an IoT environment [60]

3.1.3 Taxonomy of security protocols

The security protocols designed for IoT environment can be categorized into various types as shown in the taxonomy presented in Figure 3.1.

1) Key management

Key management in IoT can be further categorized into two types based on the type of cryptosystem used [60].

Key management based on public key cryptography is usually used to establish a secret key among two (or more) communicating entities in a network. The techniques are based on the well-known "public key based RSA algorithm" [169] or "Diffie-Hellman Key Exchange

(DHKE) algorithm" [64]. Since DHKE is insecure against "man-in-the-middle attack", its secure version ("station-to-station key agreement protocol") can be used for secure communication among entities [112]. However, both RSA and DHKE algorithms require expensive operations due to modular exponentiation operations used by those algorithms. Recently, "Elliptic Curve Cryptography (ECC)" has drawn considerable attention because of its efficiency and security as compared to the RSA algorithm. A significantly low computation time and resources required in ECC as compared to those in RSA make it suitable for resource constrained environments such as those involving IoT sensing devices and sensors.

In a symmetric key based on pre-shared keys scheme, the key pre-distribution schemes are based on the "bootstrapping protocol that establishes cryptographically symmetric keys among two communicating sensing nodes in the network". A bootstrapping protocol must not only enable a newly deployed sensing node to initiate a secure communication, but it must also permit the sensing node to join at a later time to establish secure communications with other existing sensing nodes. However, the resource limitations of the sensing nodes and the vulnerability of the physical sensing node capture attack make the implementation of this protocol a challenge. Secure network connectivity is defined by the probability of establishing a direct pairwise key between two neighbor IoT smart devices. Assume that this probability is denoted by p_{con}, where $0 \leq p_{con} \leq 1$. Now, if $p_{con} = 1$, a key pre-distribution scheme is called *deterministic*; otherwise, it will be called *probabilistic*.

2) Authentication

Authentication is an important security service in the IoT environment.

- *User authentication:* In sensitive applications (e.g., healthcare and surveillance) of IoT, real-time information is much needed to take immediate and corrective actions. Since the information collected by the GWN may not always be real-time, it is necessary to access the real-time data directly from the desired sensing nodes. Therefore, we need to design user authentication schemes in IoT.

  A typical user authentication scheme in IoT has the following phases:

  - *System setup:* It allows the system parameters to be chosen by the GWN.
  - *Sensing node registration:* Before the sensing nodes are deployed or installed, they need to be registered with the GWN. The GWN then loads the necessary secret credentials before deployment.

- *User registration:* To access information (service) from certain sensing nodes, a user U_i needs to register with the GWN. U_i first provides his/her credentials (e.g., identity, password and biometrics) secretly to the GWN and the GWN issues a smart card or mobile device securely to U_i.
- *Login:* In this phase, U_i enters his/her credentials, and these are validated by smart card (mobile device), a login request message is formed and sent to the GWN via open channel.
- *Authentication and key agreement:* After receiving the login request message, the GWN first validates it and if the validation passes, the GWN sends an authentication request message to the sensing node being accessed, say SN_j. SN_j then validates the received message and dispatches the authentication reply to U_i. U_i also validates the received message from SN_j. Only after mutual authentication between U_i and SN_j a session key SK_{ij} is established between them. Both later use SK_{ij} secure communication.
- *Password & biometric update:* This phase is needed only when a legitimate user U_i wishes to update his/her password and biometrics. It is desirable that U_i should not involve the GWN for this activity and hence, this phase can be entirely executed locally without the involvement of the GWN by U_i.
- *Smart card (mobile device) revocation:* If the smart card (mobile device) is lost or stolen by an adversary, a user authentication scheme should allow the revocation phase to issue a new smart card (mobile device) with a new set of credentials stored into it.
- *Dynamic sensing node addition:* This phase is needed when some sensing nodes are captured by an adversary or some sensing nodes are exhausted because of a power failure (if the sensing devices are battery powered).

Depending on the number of factors considered in a user authentication scheme, it is called a single-factor or a multi-factor scheme. For example, if only the user password is used, a user authentication scheme is called single-factor scheme. If a smart card (mobile device) and a user password are used, it is called a two-factor scheme, and if smart card (mobile device), user password and biometrics are used, it is called a three-factor scheme [102].

- *Device authentication:* Device authentication in IoT is useful when two IoT sensing

devices need to authenticate each other for secure communications between them.

Jang *et al.* [99] designed an efficient device authentication mechanism. Their scheme works without involving some central authority. They applied the Merkle hash-tree to achieve authentication. Sharaf-Dabbagh and Saad [181] presented an authentication mechanism for the IT environment wherein the devices apply the fingerprinting methods along with the transfer learning. Their scheme handles emulation attacks effectively by differentiating normal changes in the fingerprints due to the environment from the changes done by an attacker. Sciancalepore *et al.* [177] developed another device authentication and key management scheme. Their scheme applies the implicit certificates with the ECC Diffie-Hellman key exchange protocol. The authors showed that their scheme is energy-efficient with respect to other conventional schemes, replay attack protection, fast re-keying mechanism and robust key negotiation.

3) Access control

Here, we consider two types of access control mechanisms in the IoT environment, namely "device access control" and "user access control".

- *Device access control:* The deployment of new sensing nodes in the IoT environment is necessary to prolong the lifetime of sensing nodes because the nodes may stop functioning due to battery power or because of a physical node capture by an attacker. A deployed sensing node may not be a genuine node because malicious nodes can be deployed by an adversary. In this case, it becomes hard to distinguish malicious new nodes from genuine nodes in the IoT environment. This requires a sensing device access control method when new sensing nodes are deployed in order to prevent malicious nodes from entering the network. Such an access control method primarily deals with the two tasks: a) node authentication and b) key establishment.

- *User access control:* To provide access right only to legal users for different services, information and resources available in the IoT environment, user access control is also another crucial security mechanism. Le *et al.* [115] proposed a mutual user access control scheme based on ECC. The user access control scheme proposed by Wang *et al.* [205] also relies on ECC but it is not scalable in that it does not support a large number of sensing nodes. An ECC-based identity-based signature is applied in Mahmud-Morogan's user access control scheme [135]. Later, He *et al.* [91] developed a user access control scheme which maintains a network user to protect his/her data access privacy from the

network owner. Moreover, Chatterjee *et al.* [45] also proposed a hybrid approach with the help of both public key ECC and symmetric cryptosystems to design an efficient user access control scheme.

4) Privacy preservation

Some IoT applications contain the sensitive information collected by the IoT sensing devices. Such information needs to kept private and secure. Examples of sensitive IoT information include physiological information collected by wearable devices [58, 61] and implantable medical devices [215], secret information collected by IoT devices in smart home environment [219] and energy consumption information collected by smart devices [216]. Thus, if these types of information are leaked, it may pose serious threats including criminal activity and it may also result in serious harm or even death (in the case of patients). To mitigate these issues, the researchers have designed some privacy preserving schemes in IoT.

5) Identity management

In a distributed and dynamic network like the IoT environment, sensing devices and services can be disclosed to various threat agents which can reveal their data including personal and private identities of end users [31]. An Identity Provider (IdP) is a system that is required to generate, maintain, and manage the identities. A Service Provider (SP) is a system that provides services for users.

3.1.4 Functionality requirements for access control in IoT

The basic functionality requirements of an access control mechanism for an IoT environment are as follows.

- An access control mechanism must facilitate dynamic IoT smart device deployment in the target IoT network as the sensing devices often run out of battery or may go offline due to a hardware failure and node capture by an adversary. Thus, it is required to deploy new IoT smart devices to maintain the good health of the IoT environment.
- An access control mechanism must demand for mutual node authentication between any two neighbor IoT smart devices before establishing the pair-wise shared secret keys.
- An access control scheme must ensure secure communication with properly shared session keys establishment between any pair of nodes.

- Given that the IoT smart devices contain limited resources with respect to computation and transmission, an access control scheme must use a minimum number of messages for node authentication and must employ lightweight computations to use it in real-time applications.
- An access control mechanism must not involve the gateway nodes (base stations) for establishing pair-wise secret keys to communicate securely. This will greatly reduce the computation and communication overhead on the IoT smart devices. This will also make dynamic deployment of new smart devices more efficiently as the new nodes can establish the shared secret keys locally without communicating with the gateway nodes.

3.2 Existing access control schemes in smart grids

Musleh *et al.* [146] presented a survey work by considering several aspects, mechanisms, advantages, and also research challenges if the blockchain based solution is applied in the smart-grid environment. Furthermore, they presented frameworks that are essentials for blockchain based applications for smart grid. They also emphasized that the blockchain is appropriate for utilizing the cyber-physical layer of the smart grid. They also mentioned that the power grid will turn out to be a splendid usage of blockchain technology like other applications, such as industrial sectors.

Andoni *et al.* [15] examined several industrial and academic sources to present basics of blockchain related technologies, such as "system architectures" and "distributed consensus algorithms", crucial aspects of performance for blockchain related ecosystems. They particularly showcased some specific domains in which innovation is essential for energy system stakeholders as well as industrial entities.

Kim and Huh [109] presented a smart grid design which replies on power trading system and blockchain concept. Their design supports an architecture that can be applied for stable P2P transactions for transitional smart contact mechanisms in order to stably to trade power information of an already existing smart grid system. Wang *et al.* [207] examined various related issues in smart grid system, such as need for "removal of the centralized control and transition to a distributed architecture", privacy and security aspects, auction pricing approach at the settlement time, minimization of cost and maximization of benefit of the system.

Aitzhan and Svetinovic [12] discussed the issue of supporting the transaction security in a decentralized smart grid energy trading system, where it does not rely on a trusted third party

(TTP). Moreover, through implementation on a "token-based private decentralized energy trading system", they showed that such a system allows the peer entities to negotiate trading prices in an anonymous way and also to execute trading transactions in secure manner. To provide security and privacy at a certain level, they applied multi-signatures, blockchain and anonymous "encrypted message propagation streams".

In recent years, several authentication and access control schemes are designed in the smart grid systems [41, 47, 49, 92, 103, 122, 152, 156, 162, 174, 199, 246]. An authentication protocol designed by Nicanfar *et al.* [152] is applicable for a home network, where a smart meter can authenticate mutually with an authorized server. Li *et al.*'s mutual authentication scheme [122] designed for smart grid system provides secure communication and it relies on the Merkle hash tree concept. Another authentication scheme designed by Chan and Zhou [41] provides a "two-factor cyber-physical device authentication" in order to provide security in a smart grid system.

Qi and Chen [162] proposed an efficient authenticated key agreement mechanism for smart grid environment using the "Elliptic Curve Qu-Vanstone (ECQV) implicit certificate" as the building block. In their scheme, a mutual authentication takes place between a smart meter and a service provider and a session key is established at the end of the mutual authentication. However, their scheme does not support the blockchain solution. Chaudhry *et al.* [47] proposed an authentication mechanism for "demand response management" (DRMAS) in a smart grid edge computing based environment. In this scheme, the blockchain is not also supported.

Later, an anonymous key distribution method was suggested by Tsai and Lo [199]. Their method relies on "bilinear pairings" and ECC, which supports mutual authentication among a smart meter and a service provider. Also, a session key among them is crated for secret communication once mutual authentication is successful. Unfortunately, their scheme is not resilient to "ephemeral secret leakage (ESL)" attack, and also it does not offer "strong credentials' privacy of a smart meter" [156]. To eradicate the limitations of Tsai and Lo's scheme [199], another authentication mechanism was suggested in [156]. He *et al.* [92] also designed another "Elliptic Curve Cryptography (ECC)-based anonymous key distribution scheme" for a smart grid environment to reduce communication as well as computation overheads as compared to Tsai and Lo's protocol [199].

An authentication mechanism designed by Mahmood *et al.* [133] fails to support "anonymity" feature because they transmitted the smart meter's actual identity openly in the network. In addition, their scheme fails to prorect "known session-specific temporary information attack (ESL attack)", and also does not provide "perfect forward security", and

"private keys leakage security" [9]. Furthermore, Mahmood *et al.* [134] also suggested another authentication mechanism for smart grid system that is based on edge computing paradigm. Unfortunately, Wang *et al.* [206] indicated that mutual authentication is not achieved in the scheme [134] since the validity of utility control is not verified by a smart meter. In addition, Wang *et al.* [206] designed a mutual authentication mechanism that utilizes blockchain technology. Because of utilization of blockchain, their protocol provides conditional privacy issue and also key management.

Gai *et al.* [82] developed a model for "permissioned blockchain edge" in a smart grid system. Their designed model is able to provide privacy protection along with energy security by applying both blockchain technology and edge computing facility. They further applied group signatures and channel authorization mechanisms to assure the validity of the involved users in the smart grid.

Zhang *et al.* [239] designed a "decentralized keyless signature scheme based on a consortium blockchain" to develop an effective key management. In their approach, the smart meters send requests and then receive the responses using a P2P blockchain network for data transmission. They suggested a decentralized consensus mechanism that does not need a trusted third party or a trusted anchor. Zhou *et al.* [246] proposed an access control mechanism using blockchain in the power system, which relies on "identity-based combined encryption", "digital signature" and "signcryption". Their approach solves the "key escrow" issue of the distrustful third parties.

Recently, Yao *et al.* [228] designed a "decentralized autonomous organization (DAO) trading platform" for an industrial IoT environment that relies on blockchain network assisted with cloud mining concept. They further modeled the "computational resource management and pricing problem" along the miners and the resource provider as a "Stackelberg game". In addition, they applied a "multiagent reinforcement learning" technique in order to attain the near-optimal strategy.

Table 3.1 summarizes various existing competing authenticated key agreement schemes with respect to their cryptographic techniques used, advantages and limitations/drawbacks.

Table 3.1: Cryptographic techniques, advantages and limitations of existing authentication/access control schemes in smart grids

Scheme	Cryptographic Techniques	Advantages	Drawbacks/Limitations
Tsai and Lo [199]	* ECC * Bilinear pairings * Hash functions * Modular exponentiations	* Mutual authentication * Session key establishment	* Vulnerable to ephemeral secret leakage (ESL) attack * No strong credentials' privacy of smart meter * High computational cost * Does not support blockchain solution
Mahmood *et al.* [133]	* One-way hash functions * ECC	* Mutual authentication * Session key establishment	* Vulnerable to ESL attack * No anonymity * No perfect forward security * No private keys leakage security * Does not support blockchain solution
Mahmood *et al.* [134]	* Bilinear pairing * ECC * One-way hash functions * Modular exponentiations	* Key agreement	* No mutual authentication * Does not support blockchain solution * Vulnerable to ESL attack * High computational cost
Wang *et al.* [206]	* One-way hash functions * ECC	* Mutual authentication * Key agreement * Support blockchain solution	* No voting-based consensus mechanism for block mining in blockchain * No secure leader selection in P2P network
Zhou *et al.* [246]	* Bilinear pairings * ECC * One-way hash functions	* Support blockchain solution * Signcryption-based access control	* No secure leader selection in P2P network * Vulnerable to ESL attack * No perfect forward security * No dynamic nodes addition after initial deployment * High computational cost
Qi and Chen [162]	* ECC * One-way hash functions	* Mutual authentication * Session key agreement	* Does not support blockchain solution * No dynamic nodes addition after initial deployment
Chaudhry *et al.* [47]	* ECC * One-way hash functions	* Mutual authentication * Session key agreement	* Does not support blockchain solution

3.3 Existing access control schemes in IoT-related other enviroments

Li *et al.* [121] proposed an access control method in an "Industrial Wireless Sensor Network (IWSN)" environment. Their scheme permits a user to authorize, revoke, and authenticate for accessing real time information inside IWSN. Though their protocol supports both public

verifiability and ciphertext authenticity, but it is impractical because of heavy computational overheads due to usage of the costly bilinear pairing operations.

Bilal and Kang [132] also designed an authentication approach in WSN deployment tailored to the IoT environment. In their protocol, a sensor node can establish multiple concurrent sessions to access information securely from other sensor nodes. Unfortunately, their approach is vulnerable to "parallel-session hijacking attack". Xue *et al.* [226] suggested an access control mechanism for a smart home environment. Their scheme allows "authentication", "secure information access", and "unified storage provision" at the same time. However, the primary drawback related to this approach is that it does not provide "key agreement". Moreover, their scheme is vulnerable to "Ephemeral Secret Leakage (ESL) attack".

Li *et al.* [124] presented a three-factor user authentication scheme for IIoT environment. Unfortunately, their scheme fails to provide forward security and mobile device loss attack. Luo *et al.* [131] designed another access control mechanism for WSN-based IoT environment. Since their scheme is based on the identity based cryptographic technique, it is obviously heavy in computation due to costly bilinear pairing operations.

Li *et al.* [123] designed an elliptic curve cryptography (ECC)-based authentication scheme for IIoT which preserves privacy of the user and gateway nodes, and also provides wrong password detection mechanism quickly. Zeng *et al.* [235] designed an "anonymous user authentication (E-AUA)" protocol for both users and servers in an IoT environment. E-AUA uses multi-server environment to provide better services and also to overcome network congestion. Their scheme is also computationally expensive as costly bilinear pairing operations are applied. Moreover, their scheme is susceptible to "offline password guessing", "privileged-insider", and "server secret key leakage" attacks as mentioned in [138].

Esfahani *et al.* [73] designed an authentication protocol for IIoT with low computational cost which is based on the lightweight primitives like one-way hash function and XOR operations. However, their scheme requires to store secret authentication information on an authentication server, which may endanger a single point of failure. Garg *et al.* [84] proposed another lightweight ECC based authentication scheme for IoT based Industry 4.0 application. Though their scheme requires less computation cost, it does not resist against IoT smart device impersonation attack.

Leyou *et al.* [240] introduced a privacy preserving based CP-ABE scheme that supports authority verification without any privacy leakage which provides constant size private keys with short ciphertexts. It is also shown that the selective security under the "Decisional n-Bilinear Diffie-Hellman Exponent (n-BDHE)" computational problem with decisional linear

assumption, is achieved in this scheme.

Xu *et al.* [225] illustrated a framework for privacy-preserving ABAC system, which assures the security and privacy of the outsourced users' data stored in "Cloud Service Provider (CSP)". The framework also supports secure de-duplication which helps to eliminate redundant encrypted data in the CSP with decent communication costs. Tian *et al.* [198] introduced an ABE full privacy protection (ABE-FPP) scheme based on three stages; 1) key generation, 2) access control, and 3) partial decryption. It provides policy hidden strategy, known as hybrid-verification strategy, that reveals only attribute names and also is able to hide its values to preserve privacy during partial decryption.

Later, Gupta *et al.* [87] tried to address the access control problems in the "intelligent transportation system (ITS)" ecosystem by proposing an ABAC system. Their system uses the fine-grained policies with individualized privacy choice in order to grant/deny different activities in the smart entities. Han *et al.* [89] discussed the "role-based access control (RBAC)" that relies on the analysis using role-permissions matrices and also the implied concept of lattices. They evaluated their methodology by applying it to other substantial practical open-source systems, such as a) MediaWiki, b) Moodle, c) Joomla, and d) WordPress.

Amoon *et al.* [14] designed a "role-based reputed access control (RRAC)" scheme for protecting malicious attacks in an IoT system. Their scheme achieves two types of features, where it internally provides an "adaptive certificate based authentication" between users and resources, and it also externally trusts user communication. However, the role of IoT devices is determined separately based on reputation derived by the service provider (SP). In this scheme, precision of reputation is achieved by eliminating untrusted devices that are based on false reputation.

Lin *et al.* [125] proposed a blockchain-based secure access control protocol (BSeIn) for industry 4.0 which provides essential security features, such as "authentication", "auditability", and "confidentiality". Moreover, their scheme applies costly bilinear pairing operations that substantially increase the computational overheads. Ren *et al.* [168] designed a "blockchain-based access control scheme for edge based IIoT". In their scheme, two entities make the session key based on short and long terms secrets, and as a result, their scheme is secure against ESL attack. Since the timestamp is not applied in their scheme, a strong replay attack protection is not provided.

Guangsheng *et al.* [230] introduced a blockchain-based IoT application compatible with the "attribute-based encryption (ABE)", where the fine-grained access control is used for attributes updation. In addition, they introduced a verification scheme and showed their

solution outperforms in searching complexity and the system revokes the members when there is a direct data leakage. However, Gao *et al.* [83] proposed trustworthy "Ciphertext-Policy Attribute-Based Encryption (CP-ABE)" scheme using ciphertext-policy and attribute hiding access policy with the help of blockchain technology. They used "homomorphic ElGamal cryptosystem" in order to assure the privacy of the attributes.

Zhang *et al.* [244] proposed an "attribute-based access control (ABAC) framework for smart city application" using blockchain smart contract technology. Their scheme consists of a policy management using private Ethereum smart contracts for maintaining policies in ABAC. They computed the cost of gas consumption on Ethereum platform.

Nakamura *et al.* [149] proposed "Capability-Based Access Control (CapBAC)" scheme which stores and manages the capability tokens with local Ethereum-based implementation. However, their scheme fails to resist potential attacks. Moreover, Liu *et al.* [126] presented a CapBAC system using the blockchain technology to regulate the "dynamic identities (DIDs)" for different identities and access rights granting to IoT devices.

Das *et al.* [60] presented a detailed taxonomy of various security protocols needed in an IoT environment: a) key management, b) user and device authentication, c) device and user access control, d) intrusion detection, e) privacy preservation, and f) identity management. They discuss several security and functionality requirements, security challenges and attacks that are possible in an IoT environment. They provided a meticulous comparative study of existing IoT-related state-of-art security schemes based on discussed security and functionality features they offer. Sherali *et al.*[233] discussed various IoT-related security threats. They analyzed the current "cryptographic security standards" that are suitable for IoT smart devices and systems. Furthermore, they performed a comparative analysis on several protocol standards for IoT applications based on recent findings of the "National Institute of Standards and Technology (NIST)".

Deep *et al.* [63] reviewed the general architecture in an IoT environment. They discussed and examined various security aspect at each layer in the IoT protocol stack, such as "perception layer", "middleware layer", "network layer" and "application layer". The "perception layer" consists of single devices that are connected to a network in IoT environment. The devices are responsible for exchanging information. Some examples of the devices in this layer include "sensors", "actuators", "Zigbee", "Radio-Frequency Identification (RFID) frameworks", "Quick Response (QR) code", and "Global Positioning System (GPS) systems". The "middleware layer" is an enhancement of the network layer which performs extensive data processing and generate intelligent decisions. The "application layer" compromises of various

Table 3.2: Cryptographic techniques, advantages and limitations of existing authentication/access control schemes in IoT environment

Scheme	Cryptographic Techniques	Advantages	Drawbacks/Limitations
Li *et al.* [121]	* Elliptic Curve Cryptography (ECC) * Bilinear pairings * Hash functions * Modular exponentiation	* A user is allowed to authorize, revoke, and authenticate for accessing real time information inside IWSN * Signcryption	* Heavy computational overheads due to bilinear pairing operations * Does not support blockchain security solution
Bilal and Kang [132]	* Encryption/decryption * Hash function	* Mutual authentication * Key establishment * Establishment of multiple concurrent sessions among sensor nodes	* Vulnerable to parallel-session hijacking attack * Does not support blockchain solution
Xue *et al.* [226]	* Hash functions * Signcryptions * Encryption/decryption	* Authentication * Secure information access * Unified storage provision	* Does not provide "key agreement * Vulnerable to "Ephemeral Secret Leakage (ESL) attack under CK-adversary model
Li *et al.* [123]	* ECC * One-way hash function * Fuzzy extractor for biometric verification	* Mutual authentication * Session key agreement	* Vulnerable to "Ephemeral Secret Leakage (ESL) attack under CK-adversary model * Does not support security blockchain solution
Luo *et al.* [131]	* ECC * One-way hash function * Cross domain heterogeneous signcryption (CDHSC) * Bilinear pairings	* Authentication * Session key agreement	* Does not support dynamic sensor node addition phase * Does not support blockchain solution * Computationally costly due to bilinear pairing operations
Lin *et al.* [125]	* One-way hash function *Hashed message authentication code (HMAC) * Modular exponentiation	* Mutual authentication * Session key agreement * User anonymity * Supports blockchain solution	* Computationally costly
Zeng *et al.* [235]	* ECC * Bilinear pairings	* Online login and authentication * Password change phase	* Insecure against offline password guessing * Vulnerable to privileged-insider attack * Vulnerable to server secret key leakage * Does not support dynamic IoT device addition phase * Does not support blockchain solution
Ren *et al.* [168]	* ECC * One-way hash functions	* Mutual authentication * Session key agreement * Supports blockchain solution	* Vulnerable to replay attack * Does not support dynamic IoT device addition phase
Esfahani *et al.* [73]	* One-way hash function * Bitwise XOR operation	* Mutual authentication * Session key agreement	* Single point of failure * Does not support dynamic IoT device addition phase * Does not support blockchain solution
Garg *et al.* [84]	* ECC * Physically Unclonable Functions (PUFs)	* Mutual authentication * Session key agreement	* Vulnerable to IoT smart device impersonation attack * Does not support dynamic IoT device addition phase * Does not support blockchain solution
Li *et al.* [124]	* ECC * One-way hash function * Fuzzy extractor for biometric verification * Encryption/decryption	* Mutual authentication * Session key agreement	* Does not provide forward security * Insecure against mobile device loss attack * Does not support security blockchain solution

Table 3.3: Summary of drawbacks/limitations of existing schemes in IoT-enabled sensor networks environments

Scheme	Limitations/Drawbacks
Huang [95]	• Vulnerable to man-in-the-middle attack. • Does not support blockchain technology.
Huang [94]	• Does not allow regeneration of the finished hash chain. • Vulnerable to replay attack. • Does not support blockchain technology.
Kim and Lee [108]	• A newly joined device can easily masquerade itself in this scheme. • A legal registered device can also perform masquerading attack in this scheme. • Does not support blockchain technology.
Li *et al.* [120]	• Computational expensive due to IBC and "bilinear pairing". • Does not support blockchain technology.
Luo *et al.* [131]	• Computational expensive due to IBC and "bilinear pairing". • Does not support blockchain technology.
Aziz *et al.* [20]	• Vulnerable to ESL attack. • Does not preserve anonymity and untraceability properties. • Does not support blockchain technology.

IoT smart devices that can support personalized services to various users. The IoT smart devices are typically considered as simple, low power, and lightweight that are usually vulnerable to attacks. Moreover, they also discussed various security solutions that are suggested in different layers, such as "perception layer", "middleware layer", "network layer" and "application layer". Table 3.2 gives a comparative analysis on various cryptographic techniques, advantages and limitations of existing authentication/access control schemes in an IoT environment.

Among the security services, authentication and access control play very crucial security services for providing the security in an IoT environment. Several authentication and access control mechanisms have been proposed in IoT, sensor networks, healthcare, and other applications [24, 39, 52, 55, 113, 135, 137, 138, 155, 188, 211, 217, 218]. In the following, we only discuss the access control protocols related to IoT and wireless sensor networks (WSNs), because the IoT smart devices and sensors in IoT and WSNs respectively are resource constraint in nature. An access control scheme is categorized into two board types: certificateless and cetertificate-based.

A certificate-based access control mechanism in WSNs was proposed by Zhou *et al.*[247]. Their approach relies on the "elliptic curve cryptography (ECC)". They further used the "bootstrapping time" in order to avoid malicious sensor nodes deployment attack by an adversary. Later, a dynamic access control mechanism was also designed in WSNs by Huang [95].

This scheme relies on the existing Schnorr signature method [175] and also the sensor node expiration time. Unfortunately, Huang's scheme [95] was analyzed by Chatterjee *et al.* [44] to exhibit that the scheme [95] is vulnerable to an active attack, known as "man-in-the-middle attack". To remedy this security limitation, Chatterjee *et al.*[44] suggested an improved access control scheme which applies ECC technique and also "cryptographic one-way hash function" based on WSN-related environment.

In a "certificate-less access control" approach, there are two categories: a) "hash-chain based" and b) "hash-chainless". Huang and Liu [96] designed an access control mechanism in WSNs that uses one-way hash chaining. An access control mechanism suggested by Huang [94] was cryptanalyzed by Kim and Lee [108] to exhibit that the scheme of Huang [94] was insecure against replay attack. Furthermore, Huang's scheme [94] had other limitation where there was no way to renew an "exhausted hash chain". To withstand such limitations, another improved scheme was suggested by Kim and Lee [108], which overcame renewing "exhausted hash chain" problem. Later, two attacks (masquerade attacks from a new sensor node itself and other from a legal sensor node) were illustrated on Kim and Lee's scheme [108] by Zeng *et al.*[234]. In addition, another active attack is also pointed out by Shen *et al.* [182] on the scheme of Kim and Lee's scheme [108].

Braeken *et al.* [34] proposed an authentication protocol, called "efficient and distributed authentication protocol (eDAAAS)", that allows accessing the end-nodes in an IoT-enabled smart home scenario. Since eDAAAS relies on "symmetric cryptosystem" and "one-way cryptographic hash function", it is a lightweight protocol. Luo *et al.* [131] proposed another efficient scheme that permits access control in WSNs in the context of the IoT environment. However, this scheme is computationally heavy as it applies "Identity-Based Cryptography (IBC)" and "bilinear pairing" techniques. Li *et al.*[120] also designed an efficient access control approach in WSNs in the context of IoT environment that couples access privilege with certain users. This scheme is also computationally heavy as in the scheme of Luo *et al.* [131].

Recently, Aziz *et al.* [20] also proposed a "lightweight and compromise-resilient authentication (LCA)" scheme in an IoT environment. LCA is only based on lightweight cryptographic primitives, such as one-way hash function and bitwise XOR. It compromises various phases, like "registration (between a user and authentication server (AS))", "registration (between an IoT smart device and AS)", and "authentication and key exchange (between a user and an IoT smart device via the AS)". However, under the current *de facto* "Canetti and Krawczyk's model (CK-adversary model)" [37], their scheme is vulnerable to ESL attack as the session key construction is purely hinged on temporal (short-term) random secrets. In addition, their

scheme fails to maintain both anonymity and untraceability properties. Finally, Table 3.3 briefs the limitations of the state-of-art existing access control schemes that were designed for IoT-enabled WSNs.

3.4 Summary

In this chapter, we critically reviewed the relevant access control/authentication schemes related to smart grids and IoT-related environments. Next, we analyzed the consudered existing schemes and also provided their drawbacks/limitations, cryptographic techniques used and also advanatages.

Chapter 4

Private Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System

Now-a-days, modern power systems suffer from various challenges, for instance, ever-expanding electrical energy demand, exponential growth in renewable energy sector, adaptation of large-scale IoT devices and also several security threats posed in Cyber-Physical Systems (CPS). Another goal of IoT-enabled smart grid system is to maintain the reliability and stability of the system [146]. These challenges put in discovering the security solutions for reliable operations of the power system.

In recent years, there are several blockchain-based potential applications in smart grid domain [193]. Some of the smart grid applications using blockchain are provided below.

- **Power generation**: With the help of the blockchain, the dispatching organizations can have a full knowledge about the entire operation condition of a power grid in a real-time viewpoint. This helps the organizations to develop dispatching actions in order to maximize profits.

- **Power transmission and distribution**: The automation and control centers become the decentralized systems with the help of the blockchain technology that conquer the main challenges faced in the "‘traditional centralized systems".

- **Power consumptions**: Using the blockchain, it can manage the energy trading among the prosumers and various energy storage systems (e.g., electric vehicles)

Recently, blockchain-based solution is one of the promising technology that can be used for providing the security in the smart grid environment because of its uniqueness and also decentralized architecture. Since the communication among the users, smart meters and service providers take place via open environment, an adversary has opportunity to tamper with the data, and can perform various prospective attacks including "replay", "impersonation", "man-in-the-middle" and "ephemeral secret leakage (ESL)" attacks. Apart from these attacks, the adversary can also trace the communicated messages and therefore, anonymity and untraceability are the two main important functionality attributes that an access control security protocol should support those features. To deal with these issues, we propose a new decentralized blockchain-based access control protocol in IoT-enabled smart-grid system (DBACP-IoTSG), in which the data is collected from the smart meters securely by their respective service providers before forming the blocks and adding those blocks in the blockchain through a voting-based consensus mechanism in the P2P SP network. Because the blockchain provides transparency and immutability properties, once the block is added in the blockchain it can not be tampered by any adversary or even by legal entities in the smart grid network, and anybody can see the information stored in the block. In this work, we mainly concentrate on private blockchain because the collected data from the smart meters by the service providers are confidential as well as private. It is worth noticing that we have applied the blockchain as a service (BaaS) for secure data storage purpose. Blockchain-as-a-service (BaaS) is defined as the "third-party creation and management of cloud-based networks for companies in the business of building blockchain applications" [80].

4.1 Research contributions

The main contributions in this work are listed below:

- We propose a novel decentralized blockchain-based access control protocol in IoT-enabled smart-grid system, called DBACP-IoTSG, based on the network model provided in Figure 4.1. In DBACP-IoTSG, registration of smart meters and service providers is performed in offline mode prior to deployment in the IoT-enabled smart-grid environment. The access control phase associated with DBACP-IoTSG permits a smart meter SM_i to mutually authenticate with its associated service provider SP_j through "node authentication" process and then to establish a session key among them for secret communication through "key establishment" process. The pairwise secret keys among the service providers are used for secure consensus procedure. DBACP-IoTSG also supports

dynamic node addition mechanism.

- We then provide a detailed mechanism for a new block creation and addition in the blockchain through the Practical Byzantine Fault Tolerance (PBFT) based consensus mechanism [38].

- The proposed DBACP-IoTSG allows secure leader selection in the P2P SP network that is responsible for a block verification and addition in the blockchain using voting-based PBFT consensus algorithm.

- The proposed DBACP-IoTSG allows to preserve both anonymity and untraceability properties that are extremely needed in an IoT-enabled smart grid environment. In addition, DBACP-IoTSG also permits dynamic smart meter addition phase after initial deployment in an event that if some service providers SP_j may become faulty nodes or some smart meters SM_i may be physically compromised by an adversary. Furthermore, DBACP-IoTSG resists a crucial active attack under the present *de facto* model, known as the Canetti and Krawczyk's model (CK-adversary model) [37] (see the threat model in Section 4.2.2), known as "ephemeral secret leakage (ESL)" attack.

- Next, we apply the threat model given in Section 4.2.2 to provide a rigorous security analysis through the formal security under the broadly-accepted "Real-Or-Random (ROR) oracle model" [37], informal (non-mathematical) security analysis and also formal security verification using the widely-used AVISPA automated software tool [17] through simulation.

- We also provide the experimental results of various cryptographic primitives that are needed for comparative analysis using the widely-used "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5].

- A detailed comparative analysis among DBACP-IoTSG and other relevant protocols in smart grid environment shows that DBACP-IoTSG supports better security and provides more functionality attributes, and also requires less communication and computation costs.

- Finally, the blockchain implementation of DBACP-IoTSG has been carried out in order to measure computational time required for the varied number of blocks addition as well as the varied number of transactions per block in the blockchain.

4.2 System models

In this section, we follow the network and threat models that are applied in describing and also in analyzing the proposed scheme (DBACP-IoTSG).

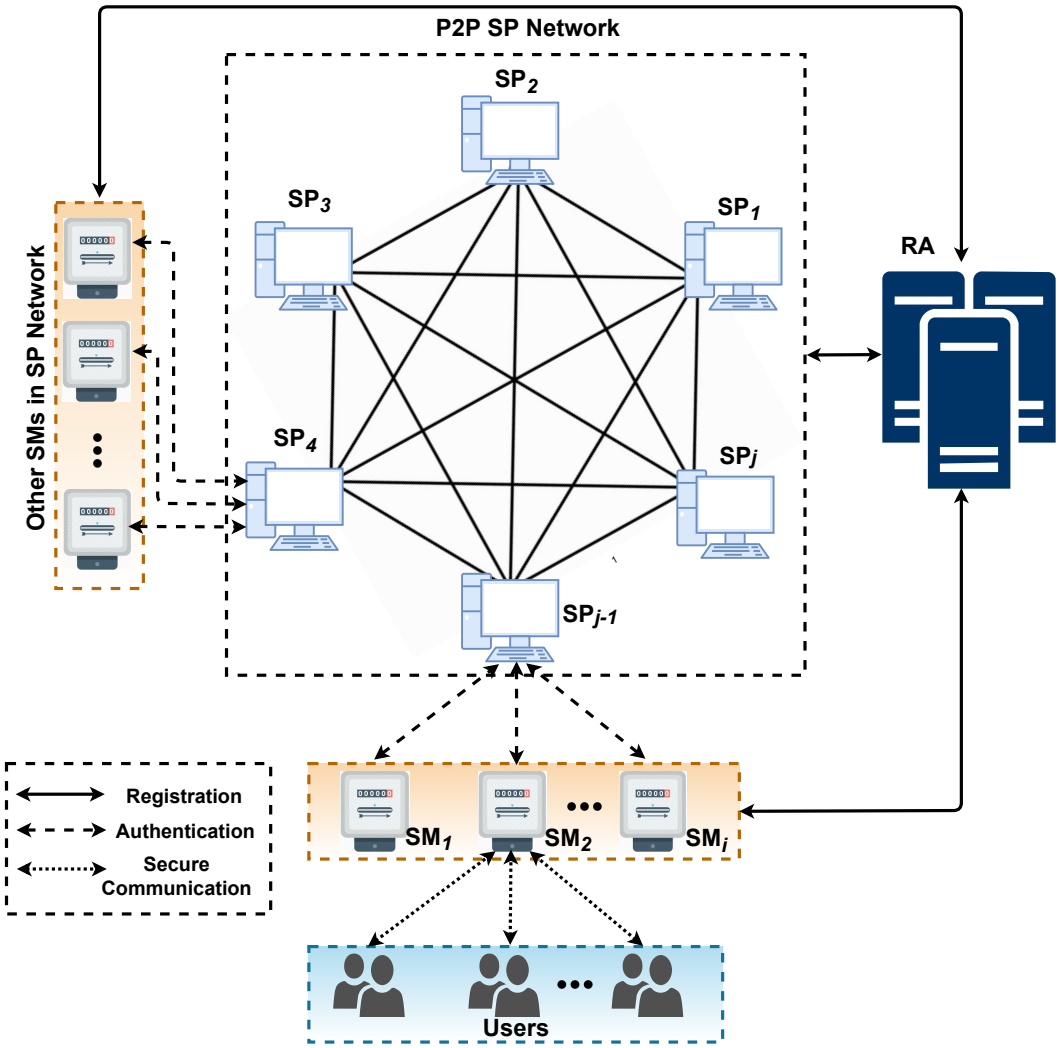

Figure 4.1: Blockchain-based IoT-enabled smart grid architecture without trusted third party (adapted from [239])

4.2.1 Network model

The network model considered in this work is shown in Figure 4.1. In this model, several users are associated with a smart meter SM_i and a group of smart meters are also associated with

a service provider SP_j. A group of service providers will form a peer-to-peer (P2P) service provider network, which is called as the P2P SP network. There is a trusted registration authority (RA) which is responsible for registering all the installed smart meters SM_i and service providers SP_j in offline mode. The RA performs the registration process securely.

The communication between the users and a smart meter SM_i takes place via secure communication, whereas a smart meter SM_i and a service provider SP_j communicate securely using a session key established among them with the help of an access control mechanism. In addition, the service providers in the SP network also establish secret pairwise keys among them for their secure communications. Under this network model, SM_i first gathers the data secretly from its associated users and then the collected data is brought secretly to the service provider SP_j under which the smart meters SM_i are registered with SP_j. SP_j then forms transactions using the collected data and creates a block. Next, the new created block can be added in the existing blockchain provided that the consensus among the service providers in the SP network is performed. Once a block is added in the blockchain, the modification or deletion of that block is not permitted to maintain "immutability" property.

4.2.2 Threat model

For our proposed decentralized blockchain-based access control protocol in IoT-enabled smart-grid system (DBACP-IoTSG), the following threat model is used.

- We contemplate the widely-accepted "Dolev-Yao (DY) threat model" [67]. According to the DY model, an adversary $\mathcal{A}$ can insert malicious information, modify or delete the message contents, apart from intercepting the messages among the communicated entities in the IoT-enabled smart grid environment.

- The end-point communicating entities (users, smart meters and service providers) are not contemplated as trustworthy parties in the network.

- We assume that some smart meters may be physically captured by $\mathcal{A}$ because the smart meters can not be monitored in 24×7. Once a smart meter is compromised physically, all the stored credentials in its memory can be extracted by $\mathcal{A}$ with the help of advanced power analysis attack [141].

- In addition, we adopt the recently contemplated *de facto* model, known as the Canetti and Krawczyk's model (CK-adversary model) [37] in the proposed DBACP-IoTSG. Under the CK-adversary model, $\mathcal{A}$ not only can intercept the messages as in the DY model,

but can also compromise secret credentials, secret keys and even session states if those information are available in insecure memory of the devices (SM_i and SP_j) during the access control phase [37].

4.3 The proposed scheme: DBACP-IoTSG

This section proposes a new decentralized blockchain-based access control protocol in IoT-enabled smart-grid system, called DBACP-IoTSG, based on the architecture shown in Figure 4.1. DBACP-IoTSG contains several phases, namely a) system setup, b) registration of smart meters and service providers, c) access control, d) key management among service providers, e) block formation and addition in the blockchain, and f) new smart meters addition after initial deployment in the smart grid environment.

To protect replay attack, we apply both random numbers and current timestamps generated by the entities in the network. It is thus assumed that the entities in the network are synchronized with their clocks. It is also a typical assumption applied in many recent authentication and access control protocols in IoT deployment [39, 42, 59, 137, 218, 219]. We use various notations tabulated in Table 4.1 for describing and also analyzing the proposed DBACP-IoTSG.

The entities involved in DBACP-IoTSG include the trusted registration authority RA, the smart meters SM_i ($i = 1, 2, \cdots, n_{sm}$) and the service providers SP_j ($j = 1, 2, \cdots, n_{sp}$), where n_{sm} and n_{sp} denote the number of smart meters and service providers to be deployed initially in the network. All the involved service providers SP_j form a peer-to-peer (P2P) blockchain network, called SPN, which are responsible for creating blocks for transactions that are securely received from their respective smart meters SM_i. A leader from the SPN is selected, which is responsible for adding the block after running the consensus algorithm.

In this chapter, we consider an access control mechanism which has basically the following two tasks [60]:

- *Node authentication:* This task demands that the newly joined nodes (SM_i and SP_j) must authenticate themselves to with other nodes to prove that they are authorized registered nodes to access the services from each other.

- *Key establishment:* This task requires that a newly deployed node needs to establish the shared secret pairwise key with its neighbor nodes after the mutual authentication

Table 4.1: Notations and their significance

Symbol	Significance
$E_q(a,b)$	A non-singular elliptic curve of the form: "$y^2 = x^3 + ax + b \pmod q$ with $4a^3 + 27b^2 \neq 0 \pmod q$"
G	A base point in $E_q(a,b)$ whose order is n as big as q
$k.G$	Elliptic curve point multiplication; $k.G = G + G + \cdots + G\,(k\,times)$
$Q + R$	Elliptic curve point addition; $Q, R \in E_q(a,b)$
$u * v$	Ordinary modular multiplication in $GF(q)$
RA, ID_{RA}	Trusted registration authority and its real identity
RID_{RA}	Pseudo-identity of the RA
mk_{RA}, Pub_{RA}	Private and public keys of RA, respectively, $Pub_{RA} = mk_{RA}.G$
SP_j	j^{th} service provider
SPN	P2P SP network of all registered service providers
ID_{SP_j}, RID_{SP_j}	Real and pseudo-identities of SP_j, respectively
TID_{SP_j}	Temporary identity of SP_j
k_{SP_j}, Pub_{SP_j}	Private and public keys of SP_j, respectively, $Pub_{SP_j} = k_{SP_j}.G$
$f(x,y)$	A symmetric bivariate t-degree polynomial over the Galois field $GF(q)$: $f(x,y) = \sum_{i=0}^{t}\sum_{j=0}^{t} a_{ij}x^i y^j$ where $a_{ij} \in Z_q = \{0,1,2,\cdots,q-1\}$
SM_i, ID_{SM_i}	i^{th} smart meter and its real identity
TID_{SM_i}, RID_{SM_i}	Temporary and pseudo identities of SM_i, respectively
TC_{SP_j}, TC_{SM_i}	Temporal credentials of SP_j and SM_i, respectively
$Cert_{SP_j}, Cert_{SM_i}$	Certificates issued by the RA to SP_j and SM_i, respectively
RTS_{SP_j}, RTS_{SM_i}	Registration timestamps of SP_j and SM_i, respectively
$\|\|$, $\oplus$	Concatenation & bitwise XOR operations, respectively
TS_x	"Current timestamp generated by an entity X" (i.e., SP_j or SM_i)
ΔT	"Maximum transmission delay associated with a message"
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"
$EC(\cdot)/DC(\cdot)$	Symmetric encryption/decryption
$EP(\cdot)/DP(\cdot)$	"Public key encryption/decryption"
MAC	"Message authentication code"

between them is done satisfactorily. The established secret keys are then used by the nodes for secret communication.

The detailed description of each phase is given below.

4.3.1 System initialization phase

The trusted RA is responsible for picking the system parameters using the following steps:

- *S1.* The RA selects a "non-singular elliptic curve of the form: $y^2 = x^3 + ax + b \pmod q$ over the Galois field $GF(q)$, where q is a large prime and $4a^3 + 27b^2 \neq 0 \pmod q$ with $\mathcal{O}$ as the point at infinity or zero point". The RA also picks a base point $G \in E_q(a, b)$ whose order will be as big as q, say n, that is, $n.G = \mathcal{O}$.

- *S2.* The RA selects an identity ID_{RA}, and picks a master key mk_{RA} as the private key and its respective public key as $Pub_{RA} = mk_{RA}.G$, and also its pseudo-identity $RID_{RA} = h(ID_{RA} \,||mk_{RA})$.

- *S3.* The RA then picks a "one-way cryptographic hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$ which takes an arbitrary length input string $x \in \{0,1\}^*$ and produces a fixed length output string of l_h bits, $h(x) \in \{0,1\}^{l_h}$". For instance, $h(\cdot)$ can be taken as "Secure Hash Algorithm (SHA-1) which produces 160-bit hash value and for more security, it can be SHA-256 or SHA-512 [140]". Moreover, to sign a message, the RA selects the "Elliptic Curve Digital Signature Algorithm (ECDSA)" [104] which contains the signature generation and verification algorithms.

- *S4.* Finally, the RA keeps mk_{RA} as its private key, and publishes other parameters $\{E_q(a, b), h(\cdot), G, Pub_{RA}\}$ which are publicly accessible to all the entities in the network.

4.3.2 Registration phase

This phase is executed by the RA in offline mode for the purpose of registering all the deployed smart meters SM_i, $(i = 1, 2, \cdots, n_{sm})$ and also the service providers SP_j, $(j = 1, 2, \cdots, n_{sp})$. It is worth noticing that the registered service providers SP_j will be part of the SPN.

1) Smart meter registration phase

The following steps are essential to complete the registration process of each deployed SM_i:

- *RSM1.* For each SM_i, the RA picks a unique real identity ID_{SM_i}, a random temporary identity TID_{SM_i} and calculates pseudo-random identity $RID_{SM_i} = h(ID_{SM_i} \,||mk_{RA}\, ||RTS_{SM_i})$, where the registration timestamp of SM_i is RTS_{SM_i}.

- *RSM2.* For each SM_i, the RA picks a random private key k_{SM_i} and calculates its respective public key $Pub_{SM_i} = k_{SM_i}.G$. In addition, the RA calculates the temporal secret for each SM_i as $TC_{SM_i} = h(ID_{SM_i} \,||mk_{RA}\, ||k_{SM_i}\, ||RTS_{SM_i})$ and certificate as $Cert_{SM_i} = k_{SM_i} + h(RID_{RA}\, ||Pub_{RA}\, ||RID_{SM_i}) * mk_{RA} \pmod q$ using its own private key mk_{RA}. The RA picks a random private key br_{SM_i} and calculates its respective public key $BPub_{SM_i} = br_{SM_i}.G$ for each SM_i.

- *RSM3.* Finally, the RA loads the credentials $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, RID_{RA}, Cert_{SM_i}, (br_{SM_i}, BPub_{SM_i})\}$ prior to its placement in the network, and declares all public keys Pub_{SM_i} as public. It is also worth noting that all the information $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, Cert_{SM_i}, br_{SM_i}\}$ are distinct for each deployed SM_i through the network. The RA deletes all the generated private keys k_{SM_i} and br_{SM_i} for each registered SM_i.

2) Service provider registration phase

Similar to the smart meter registration process, the RA proceeds to execute the following steps to complete the registration of each deployed service provider SP_j under which the smart meters SM_i, $(i = 1, 2, \cdots, n_{sm})$ will be funcional:

- *RSP1.* For each SP_j, the RA first picks a unique real identity ID_{SP_j}, a random temporary identity TID_{SP_j} and computes its pseudo-identity as $RID_{SP_j} = h(ID_{SP_j} \,||mk_{RA}\, ||RTS_{SP_j})$, where the registration timestamp of SP_j is denoted by RTS_{SP_j}.

- *RSP2.* For each registered SP_j, the RA also picks a random private key k_{SP_j} and calculates its respective public key $Pub_{SP_j} = k_{SP_j}.G$, and also the temporal secret as $TC_{SP_j} = h(ID_{SP_j} \,||mk_{RA}\, ||k_{SP_j}\, ||RTS_{SP_j})$ and certificate using its own private key mk_{RA} as $Cert_{SP_j} = k_{SP_j} + h(RID_{RA}||\, Pub_{SP_j}) * mk_{RA} \pmod q$.

- *RSP3.* For establishing pairwise secret keys among the service providers (see Section 4.3.4), we apply the "polynomial-based key distribution approach" as suggested by Blundo *et al.* [33]. To achieve this goal, the RA generates a "t-degree bivariate symmetric polynomial of the form $f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} x^i y^j$ over $GF(q)$", where the

coefficients $a_{ij} \in Z_q = \{0, 1, 2, \cdots, q-1\}$, with the property that $f(x, y) = f(y, x)$, and calculates a polynomial share for SP_j as $f(RID_{SP_j}, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} RID_{SP_j}^{i} y^j \pmod q$, which turns out to be a t-degree univariate polynomial. Note that to store this polynomial share, SP_j needs to store $(t+1)$ coefficients, which is equivalent to have storage cost of $(t+1)\log_2(q)$ bits as each coefficient is from $GF(q)$.

- *RSP4.* Finally, the RA stores the credentials $\{(TID_{SP_j}, RID_{SP_j}), TC_{SP_j}, RID_{RA}, Cert_{SP_j}, f(RID_{SP_j}, y), \{(TID_{SM_i}, RID_{SM_i}) \mid i = 1, 2, \cdots, n_{sm}\}\}$ in SP_j, deletes all the generated private keys k_{SP_j} of the registered service providers SP_j, and declares all public keys Pub_{SP_j} as public. In addition, the RA also stores $\{(TID_{SP_l}, RID_{SP_l}) \mid l \neq j, j = 1, 2, \cdots, n_{sp}\}$ in SP_j corresponding to all other service providers SP_l.

4.3.3 Access control phase

Through this phase, a smart meter SM_i will be able to authenticate with its respective service provider SP_j with the help of *node authentication* task, and then after mutual authentication between them, they will generate a common session key with the help of *key establishment* task. The detailed discussion is given through the following steps:

- *AC1.* SM_i being the initiator node generates a random secret $r_1 \in Z_q^*$ and current timestamp TS_1, and then calculates $R_1 = h(r_1 \,||RID_{SM_i}).G$, $X_1 = h(TC_{SM_i} \,||TS_1) \oplus h(RID_{RA} \,||RID_{SM_i} \,||TS_1)$ and $X_2 = h(TID_{SM_i} \,||RID_{SM_i} \,||RID_{RA} \,||R_1 \,||Cert_{SM_i} \,||TS_1)$. Next, SM_i sends the "node authentication request message" $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$ to SP_j via open channel.

- *AC2.* SP_j being the responder node, after receiving the message Msg_1 at time TS_1', first checks the validity of received TS_1 by $|TS_1 - TS_1'| < \Delta T$, and if it is valid, SP_j fetches RID_{SM_i} corresponding to received TID_{SM_i} and computes $X_2' = h(TID_{SM_i} \,||RID_{SM_i} \,||RID_{RA} \,||R_1 \,||Cert_{SM_i} \,||TS_1)$. If this criteria satisfies, SP_j further verifies the received certificate $Cert_{SM_i}$ by the condition: $Cert_{SM_i}.G = Pub_{SM_i} + h(RID_{RA} \,||Pub_{RA} \,||RID_{SM_i}).Pub_{RA}$. If it is valid, SP_j considers SM_i as valid and proceeds to the next step.

- *AC3.* SP_j calculates $h(TC_{SM_i} \,||TS_1) = X_1 \oplus h(RID_{RA} \,||RID_{SM_i} \,||TS_1)$, and generates current timestamp TS_2 and random secret $r_2 \in Z_q^*$. After this, SP_j calculates $R_2 = h(r_2 || \; RID_{SP_j}).G$, $DK_{ji} = h(r_2 || \; RID_{SP_j}).R_1$, $Y_1 = h(TC_{SP_j} \,||TS_2) \oplus h(RID_{RA} \,||RID_{SM_i} \,||TS_1 \,||TS_2)$, the session key SK_{ji} shared with SM_i as $SK_{ji} =$

$h(DK_{ji}||h(TC_{SM_i}\ ||TS_1)\ ||h(TC_{SP_j}\ ||TS_2)\ ||Cert_{SM_i}\ ||Cert_{SP_j})$. Next, SP_j generates a new temporary identity $TID^{new}_{SM_i}$ for SM_i, and calculates the session key verifier $Y_2 = h(SK_{ji}\ ||TID^{new}_{SM_i}\ ||RID_{SM_i}\ ||R_2\ ||Cert_{SP_j}\ ||TS_2)$ and $TID^{*}_{SM_i} = TID^{new}_{SM_i} \oplus h(TID_{SM_i}\ ||SK_{ji}\ ||RID_{SM_i}\ ||TS_2)$, and then sends the "node authentication response message" $Msg_2 = \{TID^{*}_{SM_i}, R_2, Y_1, Y_2, Cert_{SP_j}, TS_2\}$ to SM_i via open channel.

- *AC4.* Let SM_i receive the message Msg_2 at time TS'_2. SM_i validates the received timestamp TS_2 by $|TS_2 - TS'_2| < \Delta T$. On satisfactory validation, SM_i verifies the received certificate $Cert_{SP_j}$ by $Cert_{SP_j}.G = Pub_{SP_j} + h(RID_{RA}||\ Pub_{SP_j}).Pub_{RA}$, and if it is valid, SM_i calculates $h(TC_{SP_j}\ ||TS_2) = Y_1 \oplus h(RID_{RA}\ ||RID_{SM_i}\ ||TS_1\ ||TS_2)$, $DK_{ij} = h(r_1||\ RID_{SM_i}).R_2$, the session key SK_{ij} shared with SP_j as $SK_{ij} = h(DK_{ij}||h(TC_{SM_i}\ ||TS_1)\ ||h(TC_{SP_j}\ ||TS_2)\ ||Cert_{SM_i}\ ||Cert_{SP_j})$, $TID^{new}_{SM_i} = TID^{*}_{SM_i} \oplus h(TID_{SM_i}\ ||SK_{ij}\ ||RID_{SM_i}\ ||TS_2)$ and session key verifier $Y'_2 = h(SK_{ij}\ ||TID^{new}_{SM_i}\ ||RID_{SM_i}\ ||R_2\ ||Cert_{SP_j}\ ||TS_2)$. If the verification $Y'_2 = Y_2$ holds, SM_i generates current timestamp TS_3, calculates the session key verifier for SP_j as $X_3 = h(SK_{ij}\ ||TS_2\ ||TS_3)$ and sends the "key establishment acknowledgement message" $Msg_3 = \{X_3, TS_3\}$ to SP_j via public medium.

- *AC5.* If the message Msg_3 is received at time TS'_3, SP_j verifies the validity of TS_3 by the condition: $|TS_3 - TS'_3| < \Delta T$. Upon satisfactory verification, SP_j calculates $X'_3 = h(SK_{ji}\ ||TS_2\ ||TS_3)$ using its previous computed SK_{ji} and generated TS_2. Now, if $X'_3 = X_3$, SP_j assures that SP_j is sharing the same session key with SM_i and updates TID_{SM_i} with new $TID^{new}_{SM_i}$ in its database corresponding to SM_i. In addition, SP_j also generates a current timestamp TS_4 and computes $X_4 = h(SK_{ji}\ ||R_2\ ||TS_4)$. SP_j then constructs a message $Msg_4 = \{X_4, TS_4\}$ and sends it to SM_i via open channel.

- *AC6.* Assume that SM_i receives the message Msg_4 at time TS'_4. SM_i validates timestamp TS_4 by $|TS_4 - TS'_4| < \Delta T$. If it is validated successfully, SM_i checks if $X_4 = h(SK_{ij}\ ||R_2\ ||TS_4)$. If it is valid, SM_i assures that TID_{SM_i} has been successfully updated with $TID^{new}_{SM_i}$ at SP_j, and then replaces TID_{SM_i} with $TID^{new}_{SM_i}$ in its database too. In this way, SM_i shares the same session key SK_{ij} $(= SK_{ji})$ with SP_j.

It is worth noticing that if the message Msg_3 is lost somehow during the communication, because of sending additional message Msg_4 by SP_j the smart meter SM_i will not update TID_{SM_i} with $TID^{new}_{SM_i}$ in its database. Thus, this will solve the de-synchronization in updating temporary identity in each session between SM_i and SP_j. The overall access control phase is briefed in Figure 4.2.

Smart meter (SM_i)	Service provider (SP_j)
Generate random secret $r_1 \in Z_q^*$,	
current timestamp TS_1.	Check if $\|TS_1 - TS_1'\| < \Delta T$?
Compute $R_1 = h(r_1 \|\|RID_{SM_i}).G$,	If so, fetch RID_{SM_i} corresponding
$X_1 = h(TC_{SM_i} \|\|TS_1) \oplus h(RID_{RA}$	to TID_{SM_i}.
$\|\|RID_{SM_i} \|\|TS_1)$,	Compute $X_2' = h(TID_{SM_i} \|\|RID_{SM_i}$
$X_2 = h(TID_{SM_i} \|\|RID_{SM_i} \|\|RID_{RA}$	$\|\|RID_{RA} \|\|R_1 \|\|Cert_{SM_i} \|\|TS_1)$.
$\|\|R_1 \|\|Cert_{SM_i} \|\|TS_1)$.	Verify certificate: if $Cert_{SM_i}.G = Pub_{SM_i}$
$Msg_1 = \{TID_{SM_i}, X_1, R_1,$	$+h(RID_{RA} \|\|Pub_{RA} \|\|RID_{SM_i}).Pub_{RA}$?
$X_2, Cert_{SM_i}, TS_1\}$ $\xrightarrow{\quad}$	If so, compute $h(TC_{SM_i} \|\|TS_1) =$
(via open channel)	$X_1 \oplus h(RID_{RA} \|\|RID_{SM_i} \|\|TS_1)$.
	Generate random secret $r_2 \in Z_q^*$,
	current timestamp TS_2.
	Compute $R_2 = h(r_2 \|\| RID_{SP_j}).G$,
	$DK_{ji} = h(r_2 \|\| RID_{SP_j}).R_1$,
	$Y_1 = h(TC_{SP_j} \|\|TS_2) \oplus h(RID_{RA}$
	$\|\|RID_{SM_i} \|\|TS_1 \|\|TS_2)$,
Check if $\|TS_2 - TS_2'\| < \Delta T$?	$SK_{ji} = h(DK_{ji} \|\| h(TC_{SM_i} \|\|TS_1)$
Verify certificate: if $Cert_{SP_j}.G = Pub_{SP_j} +$	$\|\|h(TC_{SP_j} \|\|TS_2) \|\|Cert_{SM_i} \|\|Cert_{SP_j})$.
$h(RID_{RA} \|\| Pub_{SP_j}).Pub_{RA}$?	Generate new temporary identity $TID_{SM_i}^{new}$.
If so, compute $h(TC_{SP_j} \|\|TS_2) =$	Compute $Y_2 = h(SK_{ji} \|\|TID_{SM_i}^{new}$
$Y_1 \oplus h(RID_{RA} \|\|RID_{SM_i} \|\|TS_1 \|\|TS_2)$,	$\|\|RID_{SM_i} \|\|R_2 \|\|Cert_{SP_j} \|\|TS_2)$,
$DK_{ij} = h(r_1 \|\| RID_{SM_i}).R_2$,	$TID_{SM_i}^* = TID_{SM_i}^{new} \oplus h(TID_{SM_i}$
$SK_{ij} = h(DK_{ij} \|\| h(TC_{SM_i} \|\|TS_1)$.	$\|\|SK_{ji} \|\|RID_{SM_i} \|\|TS_2)$.
$\|\|h(TC_{SP_j} \|\|TS_2) \|\|Cert_{SM_i} \|\|Cert_{SP_j})$,	$Msg_2 = \{TID_{SM_i}^*, R_2,$
$TID_{SM_i}^{new} = TID_{SM_i}^* \oplus$	$Y_1, Y_2, Cert_{SP_j}, TS_2\}$ $\xleftarrow{\quad}$
$h(TID_{SM_i} \|\|SK_{ij} \|\|RID_{SM_i} \|\|TS_2)$,	(via open channel)
$Y_2' = h(SK_{ij} \|\|TID_{SM_i}^{new} \|\|RID_{SM_i}$	
$\|\|R_2 \|\|Cert_{SP_j} \|\|TS_2)$.	
If $Y_2' = Y_2$, generate current timestamp TS_3	
and compute $X_3 = h(SK_{ij} \|\|TS_2 \|\|TS_3)$.	Check if $\|TS_3 - TS_3'\| < \Delta T$?
$Msg_3 = \{X_3, TS_3\}$ $\xrightarrow{\quad}$	If so, compute $X_3' = h(SK_{ji} \|\|TS_2 \|\|TS_3)$.
(via open channel)	Check if $X_3' = X_3$?
	If valid, generate a current timestamp TS_4.
Check if TS_4 by $\|TS_4 - TS_4'\| < \Delta T$?	Compute $X_4 = h(SK_{ji} \|\|R_2 \|\|TS_4)$.
If so, check if $X_4 = h(SK_{ij} \|\|R_2 \|\|TS_4)$?	$Msg_4 = \{X_4, TS_4\}$ $\xleftarrow{\quad}$
If valid, update TID_{SM_i} with new $TID_{SM_i}^{new}$.	Update TID_{SM_i} with new $TID_{SM_i}^{new}$.
Both SM_i and SP_j store the shared common session key SK_{ij} $(= SK_{ji})$.	

Figure 4.2: Summary of access control phase in DBACP-IoTSG

4.3.4 Key management phase

Assume two service providers SP_j and SP_l agree on establishing a symmetric pairwise key between them. To achieve this purpose, the following steps are required to complete:

- *KM1.* SP_j first generates current timestamp TS_{SP_j} and sends the request message $\{TID_{SP_j}, Cert_{SP_j}, TS_{SP_j}\}$ to SP_l via open channel.

- *KM2.* SP_l after receiving the request message at time $TS^*_{SP_j}$, it validates timestamp by the condition: $|TS^*_{SP_j} - TS_{SP_j}| < \Delta T$. If it is verified successfully, SP_l validates the certificate by $Cert_{SP_j}.G = Pub_{SP_j} + h(RID_{RA}|| Pub_{SP_j}).Pub_{RA}$. If both checks are valid, SP_l considers SP_j as authentic and then retrieves RID_{SP_j} corresponding to TID_{SP_j} from its database and generates current timestamp TS_{SP_l}, calculates one-time pairwise shared key with SP_j as $K_{SP_j,SP_l} = h(f(RID_{SP_l}, RID_{SP_j}) ||Cert_{SP_j} ||Cert_{SP_l} ||TS_{SP_j} ||TS_{SP_l})$ and its verifier $VK_{SP_j,SP_l} = h(K_{SP_j,SP_l} ||TS_{SP_l})$. Next, SP_l sends the response message $\{TID_{SP_l}, Cert_{SP_l}, TS_{SP_l}, VK_{SP_j,SP_l}\}$ to SP_j via open channel.

- *KM3.* Upon reception of the response message at time $TS^*_{SP_l}$, SP_j retrieves RID_{SP_l} corresponding to received TID_{SP_l} from its database, and validates timestamp by TS_{SP_l} by $|TS^*_{SP_l} - TS_{SP_l}| < \Delta T$ and the certificate $Cert_{SP_l}$ by checking the verification equation: $Cert_{SP_l}.G = Pub_{SP_l} + h(RID_{RA}|| Pub_{SP_l}).Pub_{RA}$. If both checks are valid, SP_j calculates the one-time pairwise shared secret shared with SP_l as $K_{SP_l,SP_j} = h(f(RID_{SP_j}, RID_{SP_l}) ||Cert_{SP_j} ||Cert_{SP_l} ||TS_{SP_j} ||TS_{SP_l})$ and its verifier $VK_{SP_l,SP_j} = h(K_{SP_l,SP_j} ||TS_{SP_l})$. Note that $K_{SP_l,SP_j} = K_{SP_j,SP_l}$ as $f(RID_{SP_l}, RID_{SP_j}) = f(RID_{SP_j}, RID_{SP_l})$. If $VK_{SP_l,SP_j} = VK_{SP_j,SP_l}$, SP_j treats SP_l as authentic.

After this phase termination, both SP_j and SP_l share the same pairwise secret key K_{SP_l,SP_j} $(= K_{SP_j,SP_l})$ and use it for their secret communications in future.

4.3.5 Block formation and addition phase

During the access control phase discussed in Section 4.3.4, it is worth noticing that a service provider, say SP_j and its associated smart meter SM_i establish a session key SK_{ij} $(= SK_{ji})$. Now, using this session key SK_{ij}, SP_j will collect the encrypted informations of the form $(Tx, Sign_{Tx}, BPub_{SM_i})$ from its smart meters (SM_i), where $Sign_{Tx}$ is the ECDSA signature generation algorithm [104] and $BPub_{SM_i}$ is public key of SM_i. Thus, SP_j can decrypt the encrypted information using the same session key SK_{ij}. After that, SP_j forms an encrypted

transaction by encrypting the decrypted information using its own public key Pub_{SP_j} as $EP_{Pub_{SP_j}}[Tx_1, Sign_{Tx_1}, BPub_{SM_i}]$ and puts it into a global transactions pool, say $GTrans_{pool}$, which will be available in the P2P SP network. Assume that $GTrans_{pool}$ is filled by a list of n_t encrypted transactions, say $\{EP_{Pub_{SP_j}}(Tx_1, Sign_{Tx_1}, BPub_{SM_i}), EP_{Pub_{SP_j}}(Tx_2, Sign_{Tx_2}, BPub_{SM_i}), \cdots, EP_{Pub_{SP_j}}(Tx_{n_t}, Sign_{Tx_{n_t}}, BPub_{SM_i})\}$. Now, when $GTrans_{pool}$ reaches to the transaction threshold, say Tx_{thresh} (the minimum number of transactions (n_t) to be stored in a block $Block_m$), that is, $Tx_{thresh} = n_t$, a leader (L) will be elected by Algorithm 4 from the P2P SPN using the similar strategy mentioned in [239]. After that L will create a block $Block_m$ as mentioned in Figure 4.3.

Block Header	
Block Version (BV_m)	Unique block version number
Previous Block Hash ($PBHash_m$)	Hash value of previous block $Block_{m-1}$
Merkle Tree Root (MTR_m)	Merkle tree root on encrypted transactions
Timestamp (TS_m)	Block creation time
Owner (O_m) of $Block_m$	A service provider (SP_j)
Public key of signer O_m	Pub_{SP_j}
Block Payload (Encrypted Transactions)	
Encrypted Transactions Tx_i	$EP_{Pub_{SP_j}}(Tx_i, Sign_{Tx_i}, BPub_{SM_i})$ $(i = 1, 2, \ldots, n_t)$
Current Block Hash ($CBHash_m$)	Hash value of current block $Block_m$ acts as a signature on block
ECDSA signature on ($CBHash_m$)	$Sign_{CBHash_m}$

Figure 4.3: Formation of a block $Block_m$ on encrypted transactions by SP_j in DBACP-IoTSG

Once the block $Block_m$ is formed by the leader L, a voting-based consensus using PBFT algorithm [38] will be executed in DBACP-IoTSG for block addition in the blockchain. First of all, the leader L sends the block $Block_m$ along with a distinct random number to other service providers SP_j in the P2P SPN for consensus purpose of verifying the block. If a service provider SP_j verifies the block successfully with the existing $GTrans_{pool}$, it sends its verification status ($VerStatus$) securely to the leader L. L maintains the global commitment message pool (GCM_{pool}) which contains the valid block verification status ($VerStatus$) and it is accessible to all the peer nodes. Now, based on valid $VerStatus$, the leader L increments

its counter ($VBCount$), where $VBCount$ is the number of valid votes in the pool GCM_{pool}, which was initially set to 0. If $VBCount > 2n_f + 1$, the leader L sends *Commit* message to all the responded service providers and also adds the block $Block_m$ in the blockchain. Meanwhile, the other peer nodes in the P2P SP network will add the block in their distributed ledger. The overall process is explained in Algorithm 2. Note that the added block $Block_m$ can be verified by any service provider and also by other entities involved in the network. However, only the service provider SP_j (L) who created $Block_m$ can only decrypt the encrypted transactions containing in that block, because SP_j has the matching private key k_{SP_j} corresponding to the public key $Pub_{SP_j} = k_{SP_j}.G$ with the help of public key based ECC decryption algorithm.

Remark 4.1. *There is a pool of transactions maintained by the P2P SP network, and if it reaches to the transaction threshold, a leader (L) will be selected by the secure leader selection algorithm described in Algorithm 4. After that, the leader L will create a block and start the consensus algorithm in order to add this block into the blockchain as mentioned in Algorithm 2. To achieve this goal, the leader L will send the generated block with other encrypted random number and voting request to its all peer nodes. After getting the block, other follower service provider nodes will verify it with the existing transactions pool. Now, assume that the leader L behaves like a malicious node, generates a fake block, and then broadcasts it to the P2P SP network. After receiving the fake block, the followers will verify the block with the existing transactions pool. If the block is found to be a fake one containing the unauthorized transactions, at the verification time other follower nodes can easily verify the received block with the existing transactions pool. However, it will not be verified with the existing pool as the block contains fake transactions. Therefore, any fake block cannot be added to the blockchain. As a result, the block transparency is achieved in the proposed scheme too.*

4.3.6 Dynamic node addition phase

Sometimes, some service providers SP_j may become faulty nodes or some smart meters may be physically compromised by an adversary. Thus, it becomes essential to add some new service providers or smart meters in the existing IoT-enabled smart grid system.

Assume that a new smart meter SM_i^{new} needs to be installed under an existing service provider SP_j and a new service provider SP_j^{new} needs to be deployed in the existing P2P SP network . To achieve this goal, the RA picks for SM_i^{new} a unique real identity $ID_{SM_i}^{new}$ and a random temporary identity $TID_{SM_i}^{new}$, and computes its pseudo-random identity $RID_{SM_i}^{new} = h(ID_{SM_i}^{new} \,||mk_{RA}\, ||RTS_{SM_i}^{new})$, where the registration timestamp of SM_i^{new} is $RTS_{SM_i}^{new}$. Fur-

Algorithm 1 Secure Leader Selection

1: Suppose n_f denotes the number of faulty nodes in the SP network and $n_{sp} > 3n_f + 1$.
2: Set $Follower \leftarrow SP_j$, $(j = 1, 2, \cdots, n_{sp})$.
3: Assume N_{ov} is the number of original votes. Set $N_{ov} \leftarrow 0$.
4: Set a random timeout RT_{out} and start a timer TMR.
5: **while** $TMR > RT_{out}$ **do**
6: Set $Candidate \leftarrow Follower$.
7: Start a new timer TMR_{new}.
8: Set $N_{ov} = N_{ov} + 1$.
9: Generate a random number $r_{SP_j} \in Z_q^*$ and current timestamp TS_{SP_j}.
10: Encrypt the voting request ($VoteReq$), random number r_{SP_j} and timestamp TS_{SP_j} using the shared pairwise keys K_{SP_j,SP_l} with other service providers SP_l, $(l \neq j, j = 1, 2, \cdots, n_{sp})$ (already established during key management phase in Section 4.3.4) with the help of symmetric encryption $EC(\cdot)$ to generate the encrypted request $EC_{K_{SP_j,SP_l}}[VoteReq, r_{SP_j}, TS_{SP_j}]$.
11: Transmit the message $\{EC_{K_{SP_j,SP_l}}[VoteReq, r_{SP_j}, TS_{SP_j}], TS_{SP_j}\}$ to all other service providers SP_l and wait for the authentic votes reply messages.
12: After SP_l receives message $\{EC_{K_{SP_j,SP_l}}[VoteReq, r_{SP_j}, TS_{SP_j}], TS_{SP_j}\}$ at time $TS^*_{SP_j}$, it first checks the validity of timestamp by the condition $|TS_{SP_j} - TS^*_{SP_j}| < \Delta T$. If this condition is verified, SP_l decrypts the encrypted request using the same pairwise key K_{SP_j,SP_l} as $(VoteReq, r'_{SP_j}, TS'_{SP_j}) = DC_{K_{SP_j,SP_l}}[EC_{K_{SP_j,SP_l}}[VoteReq, r_{SP_j}, TS_{SP_j}]]$. If the decrypted timestamp TS'_{SP_j} matches with received TS_{SP_j}, SP_l sends the vote reply message $\{EC_{K_{SP_j,SP_l}}[VoteRep, r_{SP_j}, TS_{SP_l}], TS_{SP_l}\}$ to SP_j, where TS_{SP_l} is the current timestamp generated by TS_l and $VoteRep$ is the vote reply.
13: **for** each vote reply message $\{EC_{K_{SP_j,SP_l}}[VoteRep, r_{SP_j}, TS_{SP_l}], TS_{SP_l}\}$ received by SP_j from other service providers SP_l at time $TS^*_{SP_l}$ **do**
14: **if** $|TS_{SP_l} - TS^*_{SP_l}| < \Delta T$ **then**
15: Compute $(VoteRep, r^*_{SP_j}, TS^{**}_{SP_l}) = DC_{K_{SP_j,SP_l}}[EC_{K_{SP_j,SP_l}}[VoteRep, r_{SP_j}, TS_{SP_l}]]$.
16: **if** $((r^*_{SP_j} = r_{SP_j})$ and $(TS^{**}_{SP_l} = TS_{SP_l})$ and $VoteRep$ is positive) **then**
17: Set $N_{ov} = N_{ov} + 1$.
18: **end if**
19: **end if**
20: **end for**
21: **if** $N_{ov} > \frac{n_{sp}}{2} + 1$ **then**
22: $Leader \leftarrow Candidate$.
23: **else**
24: $Follower \leftarrow Candidate$.
25: Repeat Steps 7–11 for new election.
26: **end if**
27: **end while**
28: **return** $Leader$

thermore, the RA picks a random private key $k^{new}_{SM_i}$ of SM^{new}_i and calculates the respective public key $Pub^{new}_{SM_i} = k^{new}_{SM_i}.G$, SM^{new}_i's temporal secret $TC^{new}_{SM_i} = h(ID^{new}_{SM_i} \,||mk_{RA}\, ||k^{new}_{SM_i} \,||RTS^{new}_{SM_i})$ and certificate $Cert^{new}_{SM_i} = k^{new}_{SM_i} + h(RID_{RA} \,||Pub_{RA}\, ||RID^{new}_{SM_i}) * mk_{RA} \pmod q$ using its own private key mk_{RA}. The RA picks a random private key $br^{new}_{SM_i}$ and calculates its respective public key $BPub^{new}_{SM_i} = br^{new}_{SM_i}.G$ for SM^{new}_i. The RA then loads the credentials $\{TID^{new}_{SM_i}, RID^{new}_{SM_i}, TC^{new}_{SM_i}, RID_{RA}, Cert^{new}_{SM_i}, (br^{new}_{SM_i}, BPub^{new}_{SM_i})\}$ in SM^{new}_i's memory prior to its placement in the network, and declares the public keys $Pub^{new}_{SM_i}$ and $BPub^{new}_{SM_i}$ as public.

Algorithm 2 Voting-based consensus for block verification and addition in blockchain

Input: A block $Block_m$
Output: Global commitment message pool GCM_{pool} and block addition status

Assume the leader (L) has the block $Block_m$. L generates a random number rn_L and current timestamp TS_L for each follower service provider, say SP_j.
L computes the encrypted voting request ($VotReq$) using the key K_{L,SP_j} as $EC_{K_{L,SP_j}}(VotReq, rn_L)$ and $MAC_{K_{L,SP_j}}(VotReq, rn_L, TS_L)$ using the shared key K_{L,SP_j} established during the key management phase in Section 4.3.4.
L sends $Block_m$ and MAC as $\{Block_m, EC_{K_{L,SP_j}}(VotReq, rn_L), MAC_{K_{L,SP_j}}(VotReq, rn_L, TS_L), TS_L\}$ to each follower SP_j, $(j = 1, 2, \cdots, n_{sp}, L \neq SP_j)$.
Assume SP_j receives the message at time TS_L^*.
for each follower node, SP_j **do**
  if $(|TS_L - TS_L^*| < \Delta T)$ **then**
    Compute block hash $CBHash'_m$ on received $Block_m$.
    if $((CBHash'_m = CBHash_m)$ and $(Sign_{CBHash_m} = valid))$ **then**
      Compute the Merkle tree root, MTR'_m on the encrypted transactions present in the block $Block_m$.
      if $(MTR'_m = MTR_m)$ **then**
        Decrypt $VotReq$ using the key K_{L,SP_j} as $(VotReq', rn'_L) = DC_{K_{L,SP_j}}[EC_{K_{L,SP_j}}(VotReq, rn_L)]$ and MAC as $MAC_{K_{L,SP_j}}(VotReq, rn_L, TS_L)$.
        if $(MAC_{K_{L,SP_j}}(VotReq', rn'_L, TS_L) = MAC_{K_{L,SP_j}}(VotReq, rn_L, TS_L))$ **then**
          Send the block verification status $VerStatus$ as $\{EC_{K_{L,SP_j}}(rn'_L, VerStatus)\}$ to L.
        end if
      end if
    end if
  end if
end for
Initialize $VBCount \leftarrow 0$.
for each received message $\{EC_{K_{L,SP_j}}(rn'_L, VerStatus)\}$ from the follower SP_j **do**
  Compute $(rn_L^*, VerStatus) = DC_{K_{L,SP_j}}[EC_{K_{L,SP_j}}(rn'_L, VerStatus)]$.
  if $((rn_L^* = rn_L)$ and $(VerStatus = valid))$ **then**
    Set $VBCount = VBCount + 1$.
  end if
end for
if $(VBCount > 2n_f + 1)$ **then**
  Send the commit response to all followers.
  Add block $Block_m$ to the blockchain.
end if

In addition, the RA also stores the information $(TID_{SM_i}^{new}, RID_{SM_i}^{new})$ in the database of SP_j. In a similar way, the new service provider SP_j^{new} will be registered by the RA as described in Section 4.3.2 prior to its deployment.

4.4 Security analysis

This section examines the proposed DBACP-IoTSG scheme against possible attacks that are possible in blockchain-based IoT-enabled smart grid system. For this reason, we first prove the

correction of established session key between a smart meter SM_i and a service provider SP_j, and then conduct the formal security analysis under the random oracle model, the informal security analysis, and also the formal security verification to assure that DBACP-IoTSG will be secure against attacks with high probability.

4.4.1 Correctness proof of session key

Theorem 4.1 proves that the same session key is established between SM_i and SP_j.

Theorem 4.1. *The session keys established between SM_i and SP_j are the same.*

Proof. During the access control phase described in Section 4.3.3, the smart meter SM_i calculates the session key SK_{ij} shared with the service provider SP_j as $SK_{ij} = h(DK_{ij}||h(TC_{SM_i} ||TS_1) ||h(TC_{SP_j} ||TS_2) ||Cert_{SM_i} ||Cert_{SP_j})$, where $DK_{ij} = h(r_1|| RID_{SM_i}).R_2$ and $R_2 = h(r_2|| RID_{SP_j}).G$. On the other side, SP_j also calculates the session key SK_{ji} shared with SM_i as $SK_{ji} = h(DK_{ji}||h(TC_{SM_i} ||TS_1) ||h(TC_{SP_j} ||TS_2) ||Cert_{SM_i} ||Cert_{SP_j})$, where $R_1 = h(r_1 ||RID_{SM_i}).G$ and $DK_{ji} = h(r_2|| RID_{SP_j}).R_1$.

Now, it follows that

$$\begin{aligned} DK_{ij} &= h(r_1||RID_{SM_i}).R_2 \\ &= (h(r_1||RID_{SM_i}) * h(r_2||RID_{SP_j})).G \\ &= h(r_2||RID_{SP_j}).(h(r_1||RID_{SM_i}).G) \\ &= h(r_2||RID_{SP_j}).R_1 = DK_{ji}. \end{aligned}$$

Hence, $SK_{ij} = h(DK_{ij} ||h(TC_{SM_i} ||TS_1) ||h(TC_{SP_j} ||TS_2) ||Cert_{SM_i} ||Cert_{SP_j}) = h(DK_{ji} ||h(TC_{SM_i} ||TS_1) ||h(TC_{SP_j} ||TS_2) ||Cert_{SM_i} ||Cert_{SP_j}) = SK_{ji}$. □

4.4.2 Formal security under ROR model

In this section, we employ the wide-accepted Real-Or-Random (ROR) oracle model [10] to show that DBACP-IoTSG is secure against an adversary $\mathcal{A}$ for deriving the session-key between a smart meter (SM_i) and a service provider (SP_j). For this goal, we briefly discuss the ROR model with semantic security notion, and then the session key security of DBACP-IoTSG in Theorem 4.2. The adversary $\mathcal{A}$ will have the access to all the queries tabulated in Table 4.2. In addition, as discussed in [42], access to a "collision-resistant one-way cryptographic hash function $h(\cdot)$" is provided to all the involved participants including the adversary $\mathcal{A}$. As a result, we also model $h(\cdot)$ as a random oracle, say $Hash$.

Table 4.2: Queries and their purposes

Query	Purpose
$Execute(\Psi_{SM_i}^{l_1}, \Psi_{SP_j}^{l_2})$	Using this query, $\mathcal{A}$ is allowed to eavesdrop the messages exchanged between SM_i and SP_j
$CorruptSmartMeter(\Psi_{SM_i}^{l_1})$	Under this query, $\mathcal{A}$ is permitted to extract "the credentials stored in a stolen or lost SM_i"
$Reveal(\Psi^l)$	Under this query, the session key SK_{ij} $(= SK_{ji})$ shared between Ψ^l and its respective partner is leaked to $\mathcal{A}$
$Test(\Psi^l)$	Under this query, $\mathcal{A}$ is allowed to appeal Ψ^l for SK_{ij} $(= SK_{ji})$ and Ψ^l provides a "random outcome of a flipped unbiased coin, say c"

The ROR model is associated with the following components:

- **Participants.** During the access control phase, two participants, namely a smart meter (SM_i) and a service provider (SP_j) are involved apart from the RA that is only involved during the registration and dynamic node addition phases. The notations $\Psi_{SM_i}^{l_1}$ and $\Psi_{SP_j}^{l_2}$ denote the l_1 and l_2 instances of SM_i and SP_j, respectively. These instances are known as the "random oracles".

- **Accepted state.** An instance Ψ^l will enter in its "accepted state" once it goes to an accept state when the last valid protocol message is received. All the sent and received messages can be then ordered in sequence, and this constitutes the "session identification sid of Ψ^l for the current session".

- **Partnering.** Two instances (Ψ^{l_1} and Ψ^{l_2}) are partners to each other once the following three criteria are valid:

  - Ψ^{l_1} and Ψ^{l_2} need to be in "accepted states".
  - Ψ^{l_1} and Ψ^{l_2} need to share the same sid and they need to also "mutually authenticate each other".
  - Ψ^{l_1} and Ψ^{l_2} need to be "mutual partners of each other".

- **Freshness.** An instance $\Psi_{SM_i}^{l_1}$ or $\Psi_{SP_j}^{l_2}$ is called fresh when the established session key SK_{ij} $(= SK_{ji})$ shared between SM_i and SP_j is not leaked to $\mathcal{A}$ using the Reveal(Ψ^l)

query described in Table 4.2.

We now define "semantic security" of our proposed DBACP-IoTSG in Definition 4.1 prior to prove Theorem 4.2.

Definition 4.1 (Semantic security). *If we denote* $Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly})$ *as the "advantage of an adversary* $\mathcal{A}$ *running in polynomial time* t_{poly} *in breaking the semantic security of the proposed DBACP-IoTSG for computing the session key* SK_{ij} $(= SK_{ji})$ *between a smart meter* SM_i *and a service provider* SP_j*", where* $Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) = |2Pr[c' = c] - 1|$*, where* c *and* c' *are respectively the "correct" and "guessed" bits.*

Theorem 4.2. *Let there be an adversary* $\mathcal{A}$ *running in polynomial time* t_{poly} *to calculate the session key* SK_{ij} $(= SK_{ji})$ *established between a smart meter* SM_i *and a service provider* SP_j *in the proposed protocol, DBACP-IoTSG. If* q_h*,* $|Hash|$*, and* $Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$ *represent "the number of* $Hash$ *queries, the range space of a one-way collision-resistant hash function* $h(\cdot)$*, and the advantage of breaking the Elliptic Curve Decisional Diffe-Hellman Problem (ECDDHP)", respectively, then* $Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$.

Proof. We follow the proof this theorem in a similar way that was done in [24, 28, 42, 59, 137, 138, 188, 219]. There are three games, say $Game_j^{\mathcal{A}}$ for the adversary $\mathcal{A}$, $j = 0, 1, 2$, where we define $Succ_{Game_j}^{\mathcal{A}}$ as "an event that $\mathcal{A}$ can guess the random bit c in the game $Game_j^{\mathcal{A}}$ correctly". We then define $\mathcal{A}$'s advantage in winning the $Game_j^{\mathcal{A}}$ in DBACP-IoTSG as $Adv_{\mathcal{A},Game_j}^{DBACP-IoTSG} = Pr[Succ_{Game_j}^{\mathcal{A}}]$. The detailed description of each game is given below.

- $\mathbf{Game_0^{\mathcal{A}}}$: The "actual attack" performed by the adversary $\mathcal{A}$ against the proposed DBACP-IoTSG under the ROR model always corresponds the initial game $\mathbf{Game_0^{\mathcal{A}}}$. The bit c needs to be selected randomly by $\mathcal{A}$ before the game $Game_0^{\mathcal{A}}$ begins. The semantic security defined in Definition 4.1 gives the following:

$$Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) = |2Adv_{\mathcal{A},Game_0}^{DBACP-IoTSG} - 1|. \tag{4.1}$$

- $\mathbf{Game_1^{\mathcal{A}}}$: This game corresponds to an *eavesdropping game*, where the adversary $\mathcal{A}$ makes use of the defined $Execute$ query in Table 4.2. Using this query, $\mathcal{A}$ will be able to intercept all the communicated messages $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$, $Msg_2 = \{TID^*_{SM_i}, R_2, Y_1, Y_2, Cert_{SP_j}, TS_2\}$, $Msg_3 = \{X_3, TS_3\}$ and $Msg_4 = \{X_4, TS_4\}$, and try to derive the session key SK_{ij} $(= SK_{ji})$. Next, $\mathcal{A}$ needs to execute $Reveal$ and $Test$ queries in order to check whether the derived session key is a correct one or just

a random key. It is worth noting that $SK_{ij} = h(DK_{ij}||h(TC_{SM_i}||TS_1)||h(TC_{SP_j}||TS_2)||Cert_{SM_i}||Cert_{SP_j}) = h(DK_{ji}||h(TC_{SM_i}||TS_1)||h(TC_{SP_j}||TS_2)||Cert_{SM_i}||Cert_{SP_j}) = SK_{ji}$, where $DK_{ij} = h(r_1||RID_{SM_i}).R_2 = (h(r_1||RID_{SM_i}) * h(r_2||RID_{SP_j})).G = h(r_2||RID_{SP_j}).(h(r_1||RID_{SM_i}).G) = h(r_2||RID_{SP_j}).R_1 = DK_{ji}$. Since all the temporal and long term secrets are protected by $h(\cdot)$, only interception of the messages Msg_m $(m = 1, 2, 3, 4)$ will not lead to increase the success probability at all in deriving the session key SK_{ij} $(= SK_{ji})$. Now, both the games $Game_0^{\mathcal{A}}$ and $Game_1^{\mathcal{A}}$ become indistinguishable under the eavesdropping attack. Thus, we obtain the following:

$$Adv_{\mathcal{A},Game_1}^{DBACP-IoTSG} = Adv_{\mathcal{A},Game_0}^{DBACP-IoTSG}. \tag{4.2}$$

- **Game$_2^{\mathcal{A}}$**: This game will correspond to an active attack, where we include the simulations of $Hash$ and $CorruptSmartMeter$ queries, and also difficulty of solving ECDDHP. To derive the session key SK_{ij} $(= SK_{ji})$, the adversary $\mathcal{A}$ needs to derive DK_{ij} $(= DK_{ji})$, where $DK_{ij} = h(r_1||RID_{SM_i}).R_2$ and $DK_{ji} = h(r_2||RID_{SP_j}).R_1$. Assume that $\mathcal{A}$ has already the intercepted messages Msg_m $(m = 1, 2, 3, 4)$, and so, he/she has the knowledge of $R_1 = h(r_1||RID_{SM_i}).G$ and $R_2 = h(r_2||RID_{SP_j}).G$. Since $DK_{ij} = h(r_1||RID_{SM_i}).R_2 = (h(r_1||RID_{SM_i}) * h(r_2||RID_{SP_j})).G = h(r_2||RID_{SP_j}).(h(r_1||RID_{SM_i}).G) = h(r_2||RID_{SP_j}).R_1 = DK_{ji}$, the adversary $\mathcal{A}$ has to solve the computational ECDDHP to obtain DK_{ij} $(= DK_{ji})$ using R_1 and R_2. Furthermore, other secrets (TC_{SM_i} and TC_{SP_j}) are embedded in the hash function $h(\cdot)$. In addition, using the $CorruptSmartMeter$ query, the adversary $\mathcal{A}$ will have the credentials $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, RID_{RA}, Cert_{SM_i}\}$. Then, having the knowledge of other secrets, such as r_1, r_2, RID_{SP_j} and TC_{SP_j}, $\mathcal{A}$ will be able to derive the session key SK_{ij} $(= SK_{ji})$. We observe that both the games $Game_1^{\mathcal{A}}$ and $Game_2^{\mathcal{A}}$ are indistinguishable if we do not have simulation of $Hash$ and $CorruptSmartMeter$ queries, and ECDDHP is not hard problem. Using the results of birthday paradox for finding the hash collision and the advantage of solving ECDDHP, we obtain the following relation:

$$|Adv_{\mathcal{A},Game_1}^{DBACP-IoTSG} - Adv_{\mathcal{A},Game_2}^{DBACP-IoTSG}| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}). \tag{4.3}$$

It is worth noting that all the queries are made by $\mathcal{A}$, and it is only left for $\mathcal{A}$ to correctly guess a bit to win the game $Game_2^{\mathcal{A}}$. Therefore, we have,

$$Adv_{\mathcal{A},Game_2}^{DBACP-IoTSG} = \frac{1}{2}. \tag{4.4}$$

Eq. (4.1) gives

$$\frac{1}{2}.Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) = |Adv_{\mathcal{A},Game_0}^{DBACP-IoTSG} - \frac{1}{2}|. \tag{4.5}$$

Eqs. (4.2), (4.3) and (4.4), and use of triangular inequality lead to the following derivation from Eq. (4.5):

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) &= |Adv_{\mathcal{A},Game_0}^{DBACP-IoTSG} - Adv_{\mathcal{A},Game_2}^{DBACP-IoTSG}| \\ &= |Adv_{\mathcal{A},Game_1}^{DBACP-IoTSG} - Adv_{\mathcal{A},Game_2}^{DBACP-IoTSG}| \\ &\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}). \end{aligned} \tag{4.6}$$

Finally, if we multiply both sides of Eq. (4.6) by "a factor of 2", we arrive to the final result:

$$Adv_{\mathcal{A}}^{DBACP-IoTSG}(t_{poly}) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}).$$

□

4.4.3 Informal security analysis

Through the informal (non-mathematical) security analysis, we exhibit that the proposed DBACP-IoTSG resists various attacks, which are proved in Propositions 4.1–4.7.

Proposition 4.1. *DBACP-IoTSG is secure against smart meter and service provider impersonation attacks.*

Proof. We consider the following impersonation attacks related to our DBACP-IoTSG:

- **Smart meter impersonation attack:** Suppose an adversary $\mathcal{A}$ acts as a register smart meter SM_i and wants to communicate with the service provider SP_j with the message $Msg_1' = \{TID_{SM_i}, X_1', R_1', X_2', Cert_{SM_i}, TS_1'\}$. To do so, $\mathcal{A}$ can select a random number r_1' and timestamp TS_1' to calculate $R_1' = h(r_1' \,||RID_{SM_i}).G$ and $X_1' = h(TC_{SM_i} \,||TS_1') \oplus h(RID_{RA} \,||RID_{SM_i} \,||TS_1')$ and $X_2' = h(TID_{SM_i} \,||RID_{SM_i} \,||RID_{RA} \,||R_1' \,||Cert_{SM_i} \,||TS_1')$. Since RID_{SM_i} and TC_{SM_i} are the secret credentials, so without knowledge of these credentials it is "computationally infeasible problem" for $\mathcal{A}$ to generate $R_1' = h(r_1' \,||RID_{SM_i}).G$, $X_1' = h(TC_{SM_i} \,||TS_1') \oplus h(RID_{RA} \,||RID_{SM_i} \,||TS_1')$ and $X_2 = h(TID_{SM_i} \,||RID_{SM_i} \,||RID_{RA} \,||R_1' \,||Cert_{SM_i} \,||TS_1')$ on behalf of smart meter SM_i. Hence, DBACP-IoTSG is resilient against "smart meter impersonation attack".

- **Service provider impersonation attack:** Let an adversary $\mathcal{A}$ act as a valid service provider SP_j and want to send a valid message $Msg_2' = \{(TID_{SM_i}^*)', R_2', Y_1', Y_2', Cert_{SP_j}, TS_2'\}$ to SM_i. For this purpose, $\mathcal{A}$ can select a random number r_2' and timestamp TS_2' to generate $R_2' = h(r_2'||\ RID_{SP_j}).G$, $DK_{ji}' = h(r_2'||\ RID_{SP_j}).R_1$ and $Y_1' = h(TC_{SP_j}\ ||TS_2') \oplus h(RID_{RA}\ ||RID_{SM_i}\ ||TS_1'\ ||TS_2')$. $\mathcal{A}$ can also try to calculate the session key $SK_{ji}' = h(DK_{ji}'||h(TC_{SM_i}\ ||TS_1')\ ||h(TC_{SP_j}\ ||TS_2')\ ||Cert_{SM_i}\ ||Cert_{SP_j})$ and generate a new temporal identity $(TID_{SM_i}^{new})'$ to calculate session key verifier $Y_2' = h(SK_{ji}'\ ||(TID_{SM_i}^{new})'\ ||RID_{SM_i}\ ||R_2'\ ||Cert_{SP_j}\ ||TS_2')$ and $(TID_{SM_i}^*)' = (TID_{SM_i}^{new})' \oplus h(TID_{SM_i}\ ||SK_{ji}'\ ||RID_{SM_i}\ ||TS_2')$. Again it is "computationally infeasible problem" for $\mathcal{A}$ to generate R_2', DK_{ji}', Y_1', SK_{ji}', Y_2', and $(TID_{SM_i}^*)'$ without knowledge of the secrete credentials $RID_{SP_j})$, TC_{SP_j}, RID_{SM_i}, and TC_{SM_i}. Hence, DBACP-IoTSG is resilient against "service provider impersonation attack".

□

Proposition 4.2. *DBACP-IoTSG is resilient against smart meter physical capture attack.*

Proof. Due to unfriendly (unattended) environment some smart meters may be captured physically by an adversary $\mathcal{A}$. Then, $\mathcal{A}$ can extract the all stored information $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, RID_{RA}, Cert_{SM_i}\}$ from a captured smart meter SM_i which were stored during smart meter registration phase (see Section 4.3.2) using the power analysis attacks [141]. Since the secret credentials $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, Cert_{SM_i}\}$ are distinct for different smart meters, compromising these credentials can not effect to the whole network. This is primarily because these credentials will not be helpful in computing the session keys between other non-compromised smart meters and a service provider SP_j. Thus, compromise of SM_i does not reflect in compromising secure communications among non-compromised smart meters SM_i' and a service provider SP_j, and hence, they can still communicate with 100% security. This assures that DBACP-IoTSG is "unconditionally secure against smart meters capture attack" and as a result, it is protected from "smart meter physical capture attack". □

Proposition 4.3. *Replay attack is protected in DBACP-IoTSG.*

Proof. The messages $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$, $Msg_2 = \{TID_{SM_i}^*, R_2, Y_1, Y_2, Cert_{SP_j}, TS_2\}$, $Msg_3 = \{X_3, TS_3\}$ and $Msg_4 = \{X_4, TS_4\}$ are sent over public channel during the "access control phase" discussed in Section 4.3.4, which is happened in between smart meter SM_i and service provider SP_j. At the time of message creation, not only the timestamps are included but the random numbers are also attached along with the messages.

The timestamps are verified by the receiver(s) for checking message integrity and freshness. Since the adversary $\mathcal{A}$ cannot produce the original timestamps which are used in that time attached in the messages, replaying the old valid messages are then detected by the receivers. Thus, DBACP-IoTSG is protected from the "replay attack". □

Proposition 4.4. *Man-in-the-middle attack is protected in DBACP-IoTSG.*

Proof. In this attack, an adversary $\mathcal{A}$ may intercept the "node authentication request message" $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$ from an insecure (open) channel and generate another valid message Msg_1' on the fly so that the service provider SP_j as the receiver can not detect it as a modified one. But, $\mathcal{A}$ can not generate the values of R_1, X_1, and X_2 to produce the valid message Msg_1' due to the pre-loaded secrete credentials TC_{SM_i} and RID_{SM_i}. In a similar way, $\mathcal{A}$ also fails to create valid "node authentication response message" for the intercepted message $Msg_2 = \{TID^*_{SM_i}, R_2, Y_1, Y_2, Cert_{SP_j}, TS_2\}$ due to pre-loaded secret information TC_{SP_j} and RID_{SP_j}, and the shared session secrete key SK_{ji} is needed for this purpose. Furthermore, $\mathcal{A}$ can not temper the "key establishment acknowledgment message" $Msg_3 = \{X_3, TS_3\}$ and Msg_4 due to computation of the authentic secret shared session key SK_{ji}. Thus, DBACP-IoTSG is secure against the "man-in-the-middle attack". □

Proposition 4.5. *DBACP-IoTSG is resilient against ephemeral secret leakage (ESL) attack.*

Proof. In DBACP-IoTSG, using the access control phase, the smart meter SM_i calculates the session key SK_{ij} shared with the service provider SP_j as $SK_{ij} = h(DK_{ij}||h(TC_{SM_i}\ ||TS_1)\ ||h(TC_{SP_j}\ ||TS_2)\ ||Cert_{SM_i}\ ||Cert_{SP_j})$, where $DK_{ij} = h(r_1||\ RID_{SM_i}).R_2$ and $R_2 = h(r_2||\ RID_{SP_j}).G$. On the other side, SP_j also calculates the session key SK_{ji} shared with SM_i as $SK_{ji} = h(DK_{ji}||h(TC_{SM_i}\ ||TS_1)\ ||h(TC_{SP_j}\ ||TS_2)\ ||Cert_{SM_i}\ ||Cert_{SP_j})$, where $R_1 = h(r_1\ ||RID_{SM_i}).G$ and $DK_{ji} = h(r_2||\ RID_{SP_j}).R_1$. Since $DK_{ji} = DK_{ij}$, both SM_i and SP_j share the same session key $SK_{ij} = (= SK_{ji})$, which is also proved in Theorem 4.1. It is understandable that the "session key" is the combination of both the session-temporary (ephemeral) credentials (also called as "short term secrets"), such as random numbers and the "long-term secrets" (different secret credentials and pseudo-identities). The session key SK_{ij} can only be disclosed when an adversary $\mathcal{A}$ compromises both the session-temporary as well as long-term secrets. Moreover, usage of random numbers and timestamps in computation of session keys between various SM_i and SP_j over different sessions makes distinct session keys establishment among SM_i and SP_j. Even if a session key is disclosed for a specific session, it will not result in calculating the session keys over other sessions because short and

long term secrets are used. As a result, DBACP-IoTSG is secure against "session-temporary information attack" and it also preserves the "perfect forward secrecy" property. In a nutshell, DBACP-IoTSG is secure against "ESL attack". □

Proposition 4.6. *DBACP-IoTSG preserves both anonymity and untraceability functionalities.*

Proof. Suppose an adversary $\mathcal{A}$ eavesdrops the messages Msg_1, Msg_2, Msg_3 and Msg_4. Since each of the messages does not contain the real identity ID_{SM_i} of a smart meter SM_i and the real identity ID_{SP_j} of a service provider SP_j directly, $\mathcal{A}$ cannot relate who is the sender or receiver in the node authentication and key establishment session in access control phase. Therefore, "anonymity" of both smart meter and service provider is preserved in DBACP-IoTSG. Again, the parameters involved in various messages Msg_1, Msg_2, Msg_3 and Msg_4 are purely dynamic, and these are not same for any two key establishment sessions in access control phase due to usage of both random numbers and current timestamps. Hence, $\mathcal{A}$ cannot relate whether the messages exchanged between the entities over two successive sessions belong to the same user or not. This also assures "untraceability" in DBACP-IoTSG. □

Proposition 4.7. *Block verification is supported in DBACP-IoTSG.*

Proof. In DBACP-IoTSG, the block verification by the peer nodes (service providers) in the P2P SP network is based on the well-known voting-based PBFT consensus algorithm as discussed in Algorithm 2. It is worth noticing that a smart grid system is considered as an "asynchronous distribution system". Thus, a single node (peer node) failure is treated as an independent event. Suppose an adversary $\mathcal{A}$ controls some nodes in the SP network and enables them with malicious consensus algorithm. However, in DBACP-IoTSG, the leader (L) makes a decision based on responses (erroneous response or positive response or no response). As long as the L receives an adequate number of replies from the "non-failed nodes", DBACP-IoTSG assures security and activity of asynchronous system to encounter the requirements. In addition, the leader selection in DBACP-IoTSG takes place securely through the use of established pairwise secret keys among the service providers acting as peer nodes in the SP network as illustrated in Algorithm 4. As a result, there is a negligible possibility of getting corrupted responses from the non-failed authentic nodes. Thus, in DBACP-IoTSG, the block verification takes place in a secure manner. □

```
role registrationauthority (RA, SMi, SPj: agent,
        % H is one−way hash function
        H: hash_func,
        % send and receive channels under the Dolve−Yao (DY) threat model
        Snd, Rcv: channel(dy))

% Player: RA
played_by RA
def=
local State: nat,
     IDsmi, TIDsmi, RIDsmi, MKra, RTSsmi, Ksmi, TSsmi, Pubra, Certsmi: text,
     IDra,  RIDra,  Pubsmi, TCsmi, G,  TIDspj, RTSspj, Kspj, Pubspj, RIDspj: text,
     IDspj, TCspj, Certspj: text,
     F, GF: hash_func,
     SKrasmi, SKraspj: symmetric_key
const sp1, sp2 : protocol_id
init  State := 0
transition
%%% Smart meter registration phase
1. State = 0 /\ Rcv(start) =|>
  State' := 1 /\ MKra' := new() /\ Pubra' := F(MKra'.G) /\ TIDsmi' := new()
           /\ RIDra' := H(IDra.MKra')
           /\ RTSsmi' := new() /\ Ksmi' := new()
           /\ Pubsmi' := F(Ksmi'.G)
           /\ RIDsmi' := H(IDsmi.MKra'.RTSsmi')
           /\ TCsmi' := H(IDsmi.MKra'.Ksmi'.RTSsmi')
           /\ Certsmi' := GF(Ksmi'.H(RIDra'.Pubra'.RIDsmi').MKra')
%%% Send the registration message to SMi via secure channel
          /\ Snd({TIDsmi'.RIDsmi'.TCsmi'.RIDra'.Certsmi'}_SKrasmi)
%%% Service provider registration phase
          /\ TIDspj' := new() /\ RTSspj' := new() /\ Kspj' := new()
          /\ Pubspj' := F(Kspj'.G)
          /\ RIDspj' := H(IDspj.MKra'.RTSspj')
          /\ TCspj' := H(IDspj.MKra'.Kspj'.RTSspj')
          /\ Certspj' := GF(Kspj'.H(RIDspj'.Pubspj').MKra')
%%% Send the registration message to SPj via secure channel
          /\ Snd({TIDspj'.RIDspj'.TCspj'.RIDra'.Certspj'.TIDsmi'.RIDsmi'}_SKraspj)
          /\ secret({RTSsmi'.MKra'.Ksmi'}, sp1, {RA})
          /\ secret({RTSspj'.Kspj'}, sp2, {RA})
end role
```

Figure 4.4: HLSPL Specification for the role of the *RA*

4.4.4 Formal security verification: simulation study using AVISPA

This section gives the formal security verification of the proposed DBACP-IoTSG scheme using the broadly-used automated software verification tool, known as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [17]. In recent years, AVISPA-based simulation becomes reliable as it has been applied in many authentication and access control

```
role smartmeter (RA, SMi, SPj: agent,
      % H is one-way hash function
      H: hash_func,
      % send and receive channels under the Dolve-Yao (DY) threat model
      Snd, Rcv: channel(dy))

% Player: SMi
played_by SMi
def=
local State: nat,
    IDsmi, TIDsmi, RIDsmi, MKra, RTSsmi, Ksmi, TSsmi, Pubra, Certsmi: text,
    G, IDra, R1, R2, R11, TS1, TS2, X1, X2, IDspj, Kspj, RIDspj, Pubspj, TS3: text,
    RIDra, RTSspj, X3 : text,
    F, GF: hash_func,
    SKrasmi: symmetric_key
const sp1, sp2, smi_spj_r1, smi_spj_ts1, spj_smi_r2,
    spj_smi_ts2, smi_spj_ts3 : protocol_id

init  State := 0
transition
%%% Smart meter registration phase
%% Recieve registration message from the RA securely
1. State = 0 /\ Rcv({TIDsmi'.H(IDsmi.MKra'.RTSsmi').H(IDsmi.MKra'.Ksmi'.RTSsmi')
              .H(IDra.MKra').GF(Ksmi'.H(H(IDra.MKra').
              F(MKra'.G).H(IDsmi.MKra'.RTSsmi')).MKra')}_SKrasmi) =|>
  State' := 2 /\ secret({RTSsmi'.MKra'.Ksmi'}, sp1, {RA})
%%% Access control phase
          /\ R1' := new() /\ TS1' := new()
          /\ R11' := H(R1'.F(H(IDsmi.MKra'.RTSsmi').G))
          /\ X1'  := xor(H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1'),
                    H(H(IDra.MKra').H(IDsmi.MKra'.RTSsmi').TS1'))
          /\ X2' := H(TIDsmi'.H(IDsmi.MKra'.RTSsmi').H(IDra.MKra').R11'.
                    GF(Ksmi'.H(H(IDra.MKra').F(MKra'.G).
                    H(IDsmi.MKra'.RTSsmi')).MKra').TS1')
%% Send message Msg1 to SPj via public channel
         /\ Snd(TIDsmi'.X1'.R11'.X2'.GF(Ksmi'.H(H(IDra.MKra').F(MKra'.G).
              H(IDsmi.MKra'.RTSsmi')).MKra').TS1')
%% SMi has freshly generated the values r1 and TS1 for SPj that are included in Msg1
         /\ witness(SMi, SPj, smi_spj_r1, R1')
         /\ witness(SMi, SPj, smi_spj_ts1, TS1')
%%% Receive the message Msg2 from SPj via public channel
2. State = 2 /\  Rcv(F(H(R2'.H(IDspj.MKra'.RTSspj')).G).
                  xor(H(H(IDspj.MKra'.Kspj'.RTSspj').TS2'),
                       H(H(IDra.MKra').H(IDsmi.MKra'.RTSsmi').TS1'.TS2')).
                  H( H(F(H(R2'.H(IDspj.MKra'.RTSspj')).
                     H(R1'.F(H(IDsmi.MKra'.RTSsmi').G))).
                  H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1').H(H(IDspj.MKra'.Kspj'.RTSspj').TS2').
                  GF(Ksmi'.H(RIDra'.Pubra'.RIDsmi').MKra').
                  GF(Kspj'.H(RIDspj'.Pubspj').MKra')).
                  H(IDsmi.MKra'.RTSsmi').F(H(R2'.H(IDspj.MKra'.RTSspj')).G).
                  GF(Kspj'.H(RIDspj'.Pubspj').MKra').TS2').
                  GF(Kspj'.H(H(IDspj.MKra'.RTSspj').F(Kspj'.G)).MKra').TS2') =|>
  State' := 4 /\ TS3' := new()
          /\ X3' := H(H(F(H(R2'.H(IDspj.MKra'.RTSspj')).
                    H(R1'.F(H(IDsmi.MKra'.RTSsmi').G))).
                    H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1').
                    H(H(IDspj.MKra'.Kspj'.RTSspj').TS2').
                    GF(Ksmi'.H(RIDra'.Pubra'.RIDsmi').MKra').
                    GF(Kspj'.H(RIDspj'.Pubspj').MKra')).
                    TS2'.TS3')
%%% Send message Msg3 to SPj via public channel
         /\ Snd(X3'.TS3')
%% SMi has freshly generated the value TS3 for SPj that is included in Msg3
        /\ witness(SMi, SPj, smi_spj_ts3, TS3')
% SMi's acceptance of the values r2 and TS2 (message Msg2) for SMi by SPj
        /\ request(SPj, SMi, spj_smi_r2, R2')
        /\ request(SPj, SMi, spj_smi_ts2, TS2')
end role
```

Figure 4.5: HLSPL Specification for the role of smart meter SM_i

```
role serviceprovider (RA, SMi, SPj: agent,
        % H is one−way hash function
        H: hash_func,
        % send and receive channels under the Dolve−Yao (DY) threat model
        Snd, Rcv: channel(dy))

% Player: SPj
played_by SPj
def=
local State: nat,
     G, TIDspj, IDspj, MKra, RTSspj, Kspj, IDra, IDsmi, TIDsmi: text,
     RTSsmi, Ksmi, TS1, R1, R2, TS2, DKji, SKji, Y1, Pubra: text,
     R22, RIDra, RIDsmi, Pubsmi, Y2, RIDspj, Pubspj, TS3: text,
     F, GF: hash_func,
     SKraspj: symmetric_key
const sp1, sp2, smi_spj_r1, smi_spj_ts1,
     spj_smi_r2, spj_smi_ts2: protocol_id

init  State := 0
transition
%%% Service provider registration phase
1. State = 0 ∧ Rcv({TIDspj'.H(IDspj.MKra'.RTSspj').
                   H(IDspj.MKra'.Kspj'.RTSspj').
                   H(IDra.MKra').GF(Kspj'.H(H(IDspj.MKra'.RTSspj').
                   F(Kspj'.G)).MKra').
                   TIDsmi'.H(IDsmi.MKra'.RTSsmi')}_SKraspj) =|>
  State' := 3 ∧secret({RTSspj'.Kspj'}, sp2, {RA})

%%% Access control phase
%% Receive message Msg1 from SMi via public channel
2. State = 3 ∧ Rcv(TIDsmi'.xor(H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1'),
                 H(H(IDra.MKra').H(IDsmi.MKra'.RTSsmi').TS1')).
                 H(R1'.F(H(IDsmi.MKra'.RTSsmi').G)).H(TIDsmi'.
                 H(IDsmi.MKra'.RTSsmi').H(IDra.MKra').
                 H(R1'.F(H(IDsmi.MKra'.RTSsmi').G)).
                 GF(Ksmi'.H(H(IDra.MKra').F(MKra'.G).
                 H(IDsmi.MKra'.RTSsmi')).MKra').TS1').
                 GF(Ksmi'.H(H(IDra.MKra').F(MKra'.G).
                 H(IDsmi.MKra'.RTSsmi')).MKra').TS1')=|>
  State' := 5 ∧ R2' := new() ∧ TS2' := new()
           ∧ R22' := F(H(R2'.H(IDspj.MKra'.RTSspj')).G)
           ∧ DKji' := F(H(R2'.H(IDspj.MKra'.RTSspj')).
                        H(R1'.F(H(IDsmi.MKra'.RTSsmi').G)))
           ∧ Y1' := xor(H(H(IDspj.MKra'.Kspj'.RTSspj').TS2'),
                        H(H(IDra.MKra').H(IDsmi.MKra'.RTSsmi').TS1'.TS2'))
           ∧ SKji' := H(DKji'.H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1').
                        H(H(IDspj.MKra'.Kspj'.RTSspj').TS2').
                        GF(Ksmi'.H(RIDra'.Pubra'.RIDsmi').MKra').
                        GF(Kspj'.H(RIDspj'.Pubspj').MKra'))
           ∧ Y2' := H(SKji'.H(IDsmi.MKra'.RTSsmi').R22'.
                      GF(Kspj'.H(RIDspj'.Pubspj').MKra').TS2')
%%% Send message Msg2 to SMi via public channel
           ∧ Snd(R22'.Y1'.Y2'.GF(Kspj'.H(H(IDspj.MKra'.RTSspj').
                F(Kspj'.G)).MKra').TS2')
%% SPj has freshly generated r2 and TS2 for SMi that are included in Msg2
           ∧ witness(SPj, SMi, spj_smi_r2, R2')
           ∧ witness(SPj, SMi, spj_smi_ts2, TS2')
%%% Receive message Msg3 from SMi via public channel
3. State = 5 ∧ Rcv(H(H(F(H(R2'.H(IDspj.MKra'.RTSspj')).
                       H(R1'.F(H(IDsmi.MKra'.RTSsmi').G))).
                       H(H(IDsmi.MKra'.Ksmi'.RTSsmi').TS1').
                       H(H(IDspj.MKra'.Kspj'.RTSspj').TS2').
                       GF(Ksmi'.H(RIDra'.Pubra'.RIDsmi').MKra').
                       GF(Kspj'.H(RIDspj'.Pubspj').MKra')).
                       TS2'.TS3')) =|>
% SPj's acceptance of r1, TS1 (in Msg1) and TS3 (in Msg3) for SPj by SMi
  State' := 7 ∧ request(SMi, SPj, smi_spj_r1, R1')
           ∧ request(SMi, SPj, smi_spj_r1,  TS1')
           ∧ request(SMi, SPj, smi_spj_ts3, TS3')
end role
```

Figure 4.6: HLSPL Specification for the role of service provider SP_j

```
role session (RA, SMi, SPj: agent,
              H: hash_func)
def=
  local  Sn1, Sn2, Sn3, Rv1, Rv2, Rv3: channel (dy)
  composition
     registrationauthority (RA, SMi, SPj, H, Sn1, Rv1)
   /\ smartmeter (RA, SMi, SPj, H, Sn2, Rv2)
   /\ serviceprovider (RA, SMi, SPj, H, Sn3, Rv3)
end role

%%% Role for the goal and environment
role environment()
def=
  const ra, smi, spj: agent,
        h, f, gf: hash_func,
        ts1, ts2, ts3: text,
        sp1, sp2, smi_spj_r1, smi_spj_ts1, spj_smi_r2,
        spj_smi_ts2, smi_spj_ts3 : protocol_id
  intruder_knowledge = {ra, smi, spj, h, f, gf, ts1, ts2, ts3}
  composition
          session(ra, smi, spj, h)
       /\ session(ra, i, spj, h)
       /\ session(ra, smi, i, h)
end role

goal
%%% Confidentiality (privacy)
  secrecy_of sp1, sp2

%%% Authentication
authentication_on smi_spj_r1, smi_spj_ts1, smi_spj_ts3
authentication_on spj_smi_r2, spj_smi_ts2

end goal
environment()
```

Figure 4.7: HLSPL Specification for the role of the session, goal and environment

protocols, such as the schemes in [39, 59, 137, 218, 219].

AVISPA is a "push button tool for formal verification", which can detect whether a security protocol is "safe", "unsafe" or "inconclusive" against passive and active attacks. At present, AVISPA has the ability to detect replay and man-in-the-middle attacks by analyzing a security protocol through the formal verification. To implemented a tested security protocol, the protocol needs to implement using the "High-Level Protocol Specification Language (HLPSL)". The HLPSL code written in a file with extension *·hlpsl* is translated into the "Intermediate Format (IF)" using the "HLPSL2IF" translator. The IF is then provided into one of the available four backends, namely: "On-the-fly mode-checker (OFMC)", "Constraint-logic-based Attack Searcher (CL-AtSe)", "SAT-based Model Checker (SATMC)" and "Tree Automata based on Automatic Approximations for the Analysis of Security Pro-

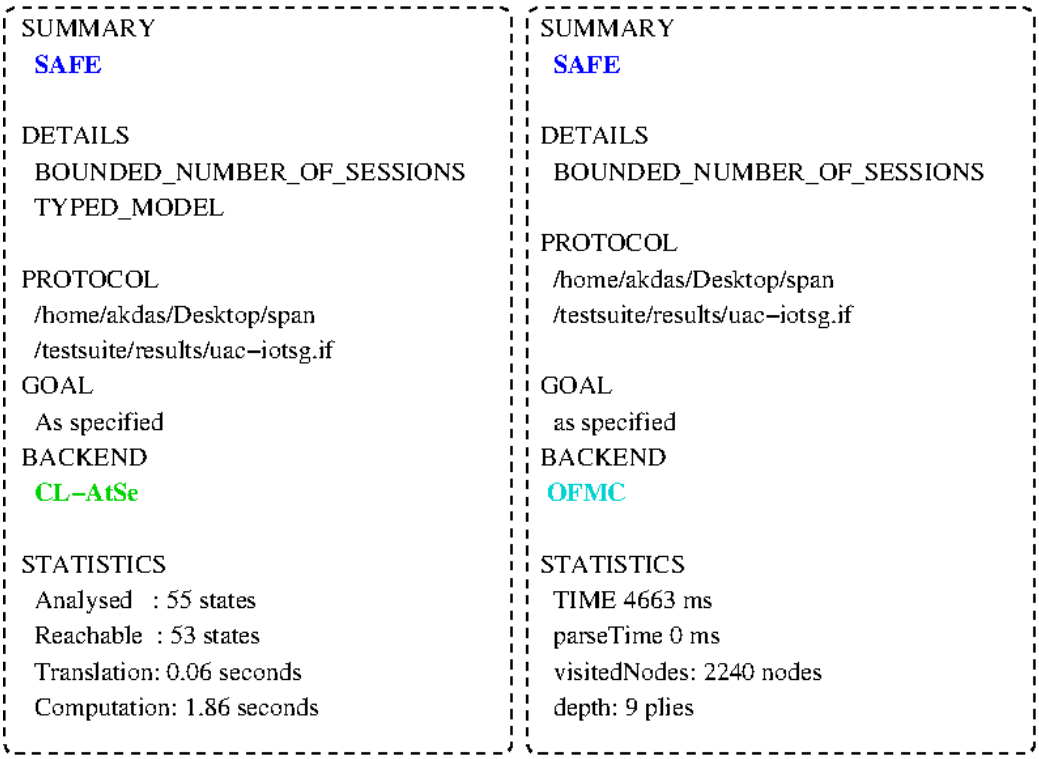

Figure 4.8: Simulation results of DBACP-IoTSG under CL-AtSe and OFMC backends

tocols (TA4SP)". Currently, SATMC and TA4SP backends do not support "bitwise exclusive OR (XOR)" operation. Therefore, in this simulation, we only stick on two backends: OFMC and CL-AtSe. The interested readers can see all the details about AVISPA and HLPSL implementation in [17].

In our implementation, we have mainly three basic roles for the RA (see in Figure 4.4), a smart meter SM_i (see in Figure 4.5) and a service provider SP_j (see in Figure 4.6). For instance, consider the HLPSL implementation for the role of a smart meter SM_i as shown in Figure 4.5. In this role, the smart meter SM_i as the initiator, during the registration phase, it receives securely the registration information $\{TID_{SM_i}, RID_{SM_i}, TC_{SM_i}, Cert_{SM_i}, br_{SM_i}\}$ generated by the RA on behalf of SM_i. During the access control phase, SM_i sends the "node authentication request message" $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$ to SP_j via open channel, with the help of

"Snd(TIDsmi'.X1'.R11'.X2'.GF(Ksmi'.H(H(IDra.MKra').F(MKra'.G).
H(IDsmi.MKra'.RTSsmi')).MKra').TS1')".

Next, after receiving the "node authentication response message" $Msg_2 = \{TID^*_{SM_i}, R_2, Y_1,$

Y_2, $Cert_{SP_j}$, $TS_2\}$ from SP_j via open channel, it processes the message and then sends the "key establishment acknowledgment message" $Msg_3 = \{X_3, TS_3\}$ to SP_j via public medium. In HLPSL, "secret({RTSsmi'.MKra'.Ksmi'}, sp1, {RA})" indicates that the secrets, namely RTS_{SM_i}, MK_{RA} and K_{SM_i} are kept secret only to the RA. The declaration "witness(SMi, SPj, smi_spj_ts3, TS3')" tells us that SM_i has freshly generated the timestamp value TS_3 for SP_j that is included in the message Msg_3. Two declarations: "request(SPj, SMi, spj_smi_r2, R2')" and "request(SPj, SMi, spj_smi_ts2, TS2')" indicate that SM_i's acceptance of the values r_2 and TS_2 in the message Msg_2 for SM_i by the SP_j.

Apart from these three basic roles, two mandatory roles for the session and goal & environment need to be implemented (see in Figure 4.7). The basic purpose of defining the role of goal is to keep a check on authentication of the communicated messages along with confidentiality of the secret variables defined in the proposed DBACP-IoTSG. Here, "confidentiality (privacy)" is achieved by the declaration: secrecy_of, whereas "authentication" is achieved by the declaration: authentication_on.

We have then simulated DBACP-IoTSG using the "Security Protocol ANimator for AVISPA (SPAN)" tool [18]. The simulation results are shown in Figure 4.8. It is worth noticing that AVISPA implements the "Dolev-Yao threat model (DY model)" [67]. Hence, an intruder (in AVISPA, it is always denoted by i) not only can intercept the communicated messages, but can also modify, delete or insert the malicious messages in between the communication. Under OFMC backend, the simulation required 4663 milliseconds (ms) with 2240 visited nodes and 9 plies of depth. Under CL-AtSe backend, the simulation analyzed 55 states and out of these states, 53 states were reachable, and it took translation time of 0.06 seconds and computation time of 1.86 seconds. The results in Figure 4.8 clearly indicate that the proposed DBACP-IoTSG is secure against replay and man-in-the-middle attacks.

4.5 Experimental results using MIRACL

In this section, we provide the experimental results of various cryptographic primitives that are needed for comparative analysis in Section 4.6 using the widely-used "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5]. MIRACL, a C programming based software library, is widely-accepted by the developers as the "gold standard open source SDK for elliptic curve cryptography (ECC)" [5].

We perform the following cryptographic operations using MIRACL. The symbols T_{bp}, T_{ecm}, T_{eca}, T_{ecenc}/T_{ecdec}, T_{mtp}, T_{exp} and T_h are used to denote the time required for "bilinear pairing",

Table 4.3: Experimental execution time (in milliseconds) for service provider

Primitive	Max. time (ms)	Min. time (ms)	Average time (ms)
T_h	0.149	0.024	0.055
T_{exp}	0.248	0.046	0.072
T_{mtp}	0.199	0.092	0.114
T_{ecm}	2.998	0.284	0.674
T_{eca}	0.002	0.001	0.002
T_{ecenc}	5.998	0.569	1.350
T_{ecdec}	3.000	0.285	0.676
T_{bp}	7.951	4.495	4.716

Table 4.4: Execution time (in milliseconds) for smart meter (Raspberry PI 3)

Primitive	Max. time (ms)	Min. time (ms)	Average time (ms)
T_h	0.643	0.274	0.309
T_{exp}	0.071	0.037	0.039
T_{mtp}	0.406	0.381	0.385
T_{ecenc}	9.085	4.427	4.592
T_{ecdec}	4.553	2.221	2.304
T_{ecm}	4.532	2.206	2.288
T_{eca}	0.021	0.015	0.016
T_{bp}	32.79	27.606	32.084

"elliptic curve point (scalar) multiplication", "elliptic curve point addition", "elliptic curve encryption/decryption", "map to elliptic curve point", "modular exponentiation" and "one-way hash function using SHA-256 hashing algorithm", respectively. Moreover, a non-singular elliptic curve of the form: "$y^2 = x^3 + ax + b \pmod q$ with $4a^3 + 27b^2 \neq 0 \pmod q$" has been considered.

We consider the following two platforms:

- **Platform 1.** In this case, the platform is considered on the setting: "Ubuntu 18.04.4 LTS, with memory: 7.7 GiB, processor: Intel Core i7-8565U CPU @ 1.80GHz × 8, OS type: 64-bit and disk: 966.1 GB". Each experiment for a cryptographic primitive is run for 100 times. Next, we have calculated the maximum, minimum and average run-time in milliseconds required for each cryptographic primitive from these 100 runs. Table 4.3

tabulates the experimental results for the considered cryptographic primitives.

- **Platform 2.** In this case, we consider the platform on the setting: "Raspberry PI 3 B+ Rev 1.3, Ubuntu 20.04 LTS, 64- bit OS, 1.4 GHz Quad-core processor, cores 4, 1 GB RAM [6]". Similar to the above case, we have also executed each primitive for 100 runs, and from these runs the maximum, minimum and average execution time in milliseconds are computed. The experimental results for each primitive are then reported in Table 4.4.

Table 4.5: Security and functionality attributes comparison

Scheme	$SFA1$	$SFA2$	$SFA3$	$SFA4$	$SFA5$	$SFA6$	$SFA7$	$SFA8$	$SFA9$	$SFA10$	$SFA11$	$SFA12$	$SFA13$	$SFA14$	$SFA15$
Zhou *et al.* [246]	✓	✓	✓	✓	✓	×	×	×	×	✓	✓	✓	×	FBFT	×
Zhang *et al.* [239]	×	✓	✓	✓	✓	×	×	×	×	×	×	✓	×	PBFT	×
DBACP-IoTSG	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	VPBFT	✓

Note: $SFA1$: "impersonation attacks"; $SFA2$: "replay attack"; $SFA3$: "man-in-the-middle attack"; $SFA4$: "mutual authentication between a smart meter and service provider/power provider/cloud storage provider" without involvement of a trusted authority; $SFA5$: "smart meter physical capture attack"; $SFA6$: "session key-security"; $SFA7$: "ESL attack"; $SFA8$: "perfect forward secrecy"; $SFA9$: "dynamic smart meter/service provider addition phase"; $SFA10$: "anonymity preservation"; $SFA11$: "untraceability preservation"; $SFA12$: "support blockchain-based solution"; $SFA13$: "secure leader selection in the P2P SP network"; $SFA14$: "type of consensus mechanism used"; $SFA15$: "secure voting during consensus algorithm" ✓: "the scheme is secure or it supports a feature"; ×: "the scheme is insecure or it does not support a feature"; FBFT: "Federated Byzantine Fault Tolerance"; PBFT: "Practical Byzantine Fault Tolerance"; VPBFT: "Voting-based PBFT".

4.6 Comparative analysis

This section gives a detailed comparative analysis among our proposed DBACP-IoTSG and other recent schemes of Zhou *et al.* [246] and Zhang *et al.* [239]. We measure the performance comparisons of DBACP-IoTSG with other schemes [239, 246] for communication and computation costs, and also for security and functionality attributes.

4.6.1 Security and functionality attributes comparison

We compare various security and functionality attributes among our proposed DBACP-IoTSG and other relevant schemes that are also based on blockchain technology [239], [246]. Table 4.5 exhibits the comparative study on these features among the schemes. It is worth noticing that our DBACP-IoTSG provides much better security and also supports more functionality features as compared to other schemes. Zhang *et al.*'s scheme [239] does not support "anonymity" and "untraceability" features which are treated as crucial features in the IoT-enabled smart grid environment. The consensus algorithms used in Zhou *et al.*'s scheme [246] and Zhang *et al.*'s scheme [239] are FBFT and PBFT, respectively. In our DBACP-IoTSG, we have utilized the puzzle-based solution combined with PBFT in order to make consensus more secure with the pairwise keys established among the service providers.

4.6.2 Communication costs comparison

For comparative study on communication costs, it is assumed that 160-bit ECC provides the same security level while it is compared with 1024-bit RSA public key cryptosystem, and thus, an "elliptic curve point of the form $G = (G_x, G_y)$, where x and y co-ordinates of G are G_x and G_y respectively", requires $(160 + 160) = 320$ bits. Since the random numbers are selected from the finite field $GF(q)$, they are 160 bits. Furthermore, a timestamp is 32 bits and a hash value (digest) is 256 bits, if SHA-256 hash algorithm [140] is utilized for blockchain technology to provide sufficient security level, and a message size in all schemes is taken as 160 bits. Table 4.6 shows comparison of communication costs among the schemes with the number of messages and the number of bits required during the access control phase. In our DBACP-IoTSG, four exchanged messages $Msg_1 = \{TID_{SM_i}, X_1, R_1, X_2, Cert_{SM_i}, TS_1\}$, $Msg_2 = \{TID^*_{SM_i}, R_2, Y_1, Y_2, Cert_{SP_j}, TS_2\}$, $Msg_3 = \{X_3, TS_3\}$, and $Msg_4 = \{X_4, TS_4\}$ demand 1184, 1280, 288, and 288 bits, a total cost of 3040 bits. It is seen that our DBACP-IoTSG requires less communication costs as compared to the scheme designed by Zhang *et al.* [239]. Although the scheme of Zhou *et al.* [246] needs less communication cost as compared to our DBACP-IoTSG, it is acceptable because it does not also support all the security and functionality attributes (see Table 4.5) as compared to DBACP-IoTSG.

4.6.3 Computational costs comparison

We mainly consider the computation costs comparison during the access control phase among the proposed DBACP-IoTSG and other existing competing schemes. We use the experimen-

Table 4.6: Communication costs comparison

Scheme	No. of messages	No. of bits
Zhou *et al.* [246]	3	2464
Zhang *et al.* [239]	4	3328
DBACP-IoTSG	4	3040

Table 4.7: Computation costs comparison

Scheme	Smart meter/User side	Service provider/Power provider /Cloud storage provider side
Zhou *et al.* [246]	$3T_{ecm} + T_{eca} + T_{mtp}$ $+3T_{bp} + 2T_h$ ≈ 104.135 ms	$6T_{ecm} + T_{eca} + T_{mtp} + 5T_{bp}$ $+T_{exp} + 8T_h$ ≈ 28.252 ms
Zhang *et al.* [239]	$4T_h$ ≈ 1.236 ms	$T_{ecenc} + T_h$ ≈ 1.405 ms
DBACP-IoTSG	$4T_{ecm} + T_{eca} + 11T_h$ ≈ 12.567 ms	$4T_{ecm} + T_{eca} + 11T_h$ ≈ 3.303 ms

tal results reported in Table 4.3 for a service provider or a power provider or cloud storage provider. On the other side, since a smart meter or a user's mobile device or smart card is resource-constrained as compared to a server, we consider the experimental results reported in Table 4.4 for the smart meter or user. In DBACP-IoTSG, a smart meter SM_i requires computational cost of $4T_{ecm}+ T_{eca}+ 11T_h$, which is roughly 12.567 milliseconds, whereas a service provider's computational cost is $4T_{ecm}+ T_{eca}+ 11T_h$, which is roughly 3.303 milliseconds. It is evident from Table 4.7 that DBACP-IoTSG needs less computational costs as compared to Zhou *et al.*'s scheme [246]. Although Zhang *et al.*'s scheme [239] demands less computational cost as compared to both Zhou *et al.*'s scheme [246] and DBACP-IoTSG, it requires more communication cost (see Table 4.6) and it does not also support all the security and functionality attributes (see Table 4.5).

4.7 Blockchain implementation

In this section, we provide the blockchain implementation of the proposed scheme (DBACP-IoTSG). The simulations were executed on a platform having "Ubuntu 18.04, 64-bit OS with Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz, 4 GB RAM". The script was developed in the

Node.js language with VS CODE 2019 [107].

We considered that the number of peer nodes (n_{sp}) in the P2P SP network is 20. A service provider is a part of the P2P SP network, that securely collects the transactions from associated smart meters to form a global transactions pool in the network. If the number of transaction in transactions pool reaches to a pre-defined transaction threshold (the minimum number of transaction to stored in a block), a leader is selected from the network for creating a block and adding that block in the blockchain. The leader, say, L creates a block $Block_m$ which has the structure as shown in Figure 4.3, and wants to add this block into the blockchain. After executing the consensus algorithm using the "Practical Byzantine Fault Tolerance (PBFT) consensus algorithm" [38] provided in Algorithm 2, a block is finally added by the leader L in the blockchain.

We consider the block version, previous block hash, Merkle tree root, timestamp (epoch time), owner (O_m) of $Block_m$ that is owner identity, public key of O_m, block payload, current block hash ($CBHash_m$) (using SHA-256 hashing algorithm), and ECDSA signature on $CBHash_m$ are of sizes 32, 256, 256, 42, 160, 320, $640n_t$, 256, and 320 bits, respectively. Each transaction Tx_i was encrypted using ECC encryption which outputs two elliptic curve points, and as a result, an encrypted transaction requires $(320 + 320) = 640$ bits. As a result, the total size for a block $Block_m$ becomes $1642 + 640n_t$ bits.

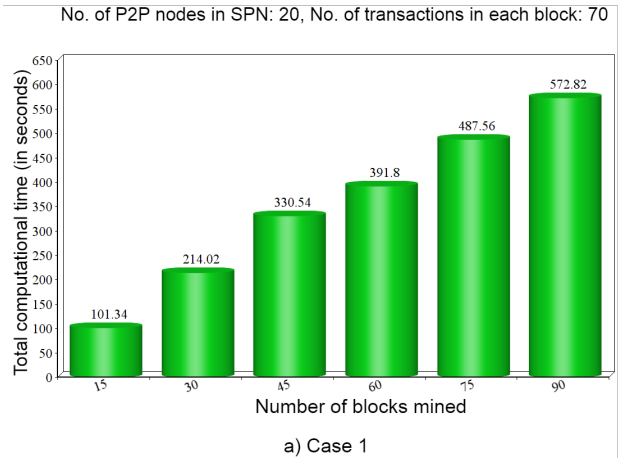

Figure 4.9: Blockchain simulation results for Case 1

In the following, three types of simulation cases are considered:

- **Case 1:** In this case, we assume $n_{sp} = 20$ and the number of transactions per block is 70. The simulation results shown in Figure 4.9 shows the number of blocks mined into

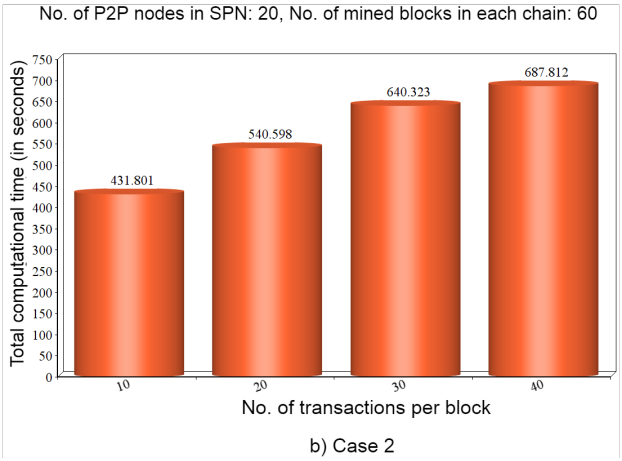

Figure 4.10: Blockchain simulation results for Case 2

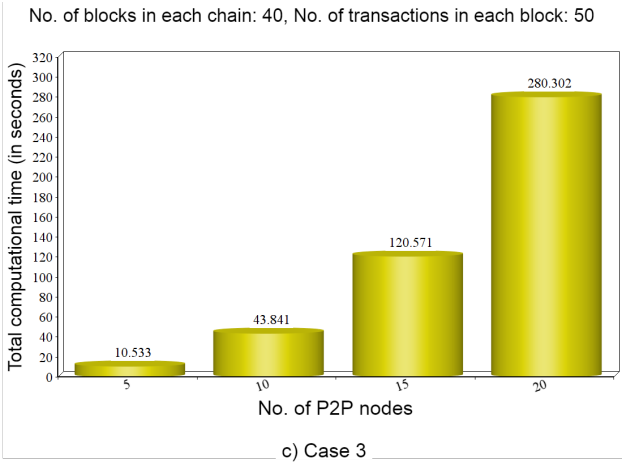

Figure 4.11: Blockchain simulation results for Case 3

the blockchain versus the total computational time (in seconds). It is noticed that as the number of blocks mined is increased, the computational time also increases.

- **Case 2:** In this scenario, we assume $n_{sp} = 20$ and the number of mined blocks in each chain is 60. The simulation results are provided in Figure 4.10. The simulation results are based on the number of transactions pushed per block versus the total computational time (in seconds). Similar to the trends observed in Case 1, the computational time also increases as the number of transactions per block is increased.

- **Case 3:** In this scenario, we have fixed the number blocks mined as 40 and the number of transactions per block as 50. The simulation results provided for Case 3 in Figure

4.11 shows the computational time increases when the number of peer nodes (n_{sp}) in the P2P network is increased.

4.8 Summary

In this work, we attempted to address an important research topic in the IoT-enabled smart grid system by designing a novel decentralized blockchain-based access control protocol in IoT-enabled smart-grid system (DBACP-IoTSG). The proposed DBACP-IoTSG works without involving a trusted third party. The blocks are then verified by a leader selection process securely in the P2P SP network. After that the leader is responsible for running the consensus algorithm securely to validate the blocks by its peer nodes using the Practical Byzantine Fault Tolerance (PBFT) method, and after successful validation the blocks are added into the blockchain. The transactions are encrypted using ECC encryption algorithm using the public key Pub_{SP_j} of a service provider SP_j who generates the blocks containing the transactions so that only SP_j can decrypt the transactions. DBACP-IoTSG is shown to be secure through a rigorous security analysis using formal, informal and simulation-based formal security verification. A thorough comparative study among the proposed DBACP-IoTSG and other relevant schemes shows DBACP-IoTSG's better security and supporting of more functionality attributes. Moreover, DBACP-IoTSG is also comparable with other schemes in terms of communication and computation costs. Finally, the blockchain based implementation for the proposed DBACP-IoTSG has been carried out to measure the computational time required for the varied number of blocks in the blockchain and also the varied number of transactions per block.

Chapter 5

Secure Access Control for Pervasive Edge Computing in Industrial IoT using Private Blockchain

In Chapter 4, we proposed an access control scheme in an Internet of Things (IoT)-enabled smart grid environment, where after secure aggregation of the data from the smart meters by their respective service providers, the transactions can be stored in blocks and then the blocks can be added into a blockchain for future analysis. Therefore, we used the blockchain as a service for secure data storage purpose only.

In this chapter, we have utilized the blockchain service not only for secure storage purpose, but also for storing the registration credentials in the blockchain. Thus, the registration credentials related to IoT smart devices, gateway nodes and edge servers during the registration process are directly stored in the blockchain center prior to their placement in the network. During the access control and key management phases, the registration credentials from the blockchain are used for authentication purposes. Recently, Pervasive Edge Computing (PEC) becomes a very emerging computing standard. PEC consists of various heterogeneous mobile edge devices, such as "smartphones", "tablets", "IoT smart devices", "gateway nodes", "edge devices", and so on. The devices can then communicate with each other to sense, process the sensing information and also to build various applications at the network edge [227]. The rapid growth of IIoT leads to many security attacks, such as "man-in-the-middle", "impersonation", "replay" and "privileged-insider" attacks, which may cause serious damage to the IIoT system. Since most traditional and existing IIoT infrastructures are based on centralized system, they are expensive, inefficient and also vulnerable to a "single-point failure". Recently,

combining blockchain technology and IoT-based security solutions achieves popularity among the researchers. Important features of blockchain (decentralization, tamper-proof, trustworthiness, traceability and immutability) provide more functionalities and greater help to the IIoT systems.

Considering the demand of large-scale IIoT systems, it becomes infeasible and inefficient to store the huge volumes of data in a traditional IIoT system. Therefore, we feel that there is a great essence in designing blockchain-based IIoT system for PEC. Thus, it should provide an efficient and robust solution to deal with the security requirements needed for PEC in IIoT environment. Since the information produced in the IIoT environment is strictly private and confidential, the information must not be leaked in public. Moreover, due to wireless communication happen among different entities in IIoT, an adversary should not be able to tamper with the sensitive data. Tampering of data may include intercepting, modifying, deleting or even inserting fake information during communication. Integrating IIoT with blockchain technology in order to develop a "secure, distributed, and stable blockchain IIoT network" seems to be a natural way [178]. In fact, the integration of blockchain along with IIoT attracted a lot of interests among the stakeholders across industry and academia as well [154]. Due to drawbacks present in IIoT-based "intelligent manufacturing system (IMS)" and also the challenging problem associated to apply the blockchain in IMS, Zhang *et al.* [236] suggested to combine IIoT with the "permissioned blockchain". In this regard, they designed an efficient "Manufacturing Blockchain of Things (MBCoT) architecture" for the configuration of a secure, decentralized, and traceable IMS. Zhang *et al.* [237] also proposed a "multi-access edge computing (MEC) enabled framework". It helps in "data security assurance" and "system latency performance" improvement.

To deal with the above-mentioned issues, we represent a novel access control scheme. It allows secure communication among IoT smart devices and their relevant gateway nodes through the designed access control process. It also provides secure communication among the gateway nodes and the connected edge servers through the designed key management process. The secure transactions are transformed into various blocks, and addition of those blocks are done through the voting-based PBFT consensus algorithm. It is done by a group of P2P edge servers nodes in the private blockchain.

5.1 Motivation

In IIoT environment, there are various types of applications connected with the system and they integrate large-scale discrete heterogeneous data. Such data can be from the smart sensor data, health care data, traffic data, environmental monitoring data, and industrial manufacturing data. In some smart energy industries, sensors, machines, and actuators collect huge amount of data such as energy, air quality, fault and resource prediction, and product planning from various locations. It further produces large data and enforces a huge amount of processing time to store the data in traditional centralized system. Moreover, in chemical industry, there is an extensive amount of critical data, such as reactivity of a catalyst in different temperature and air pressure conditions, results after the chemicals reactions. In such scenario, inefficiency of IIoT system can seriously damage the productivity of the industry. These registration credentials stored in the blockchain are then used during the access control and key management phases.

With the help of the cloud computing upgradation, IoT platform can process information in a traditional manner and transform the information into the real time actions. While the cloud storage becomes an important role in an IoT or IIoT environment, however there are issues related to threat of data, transparency and privacy preservation. This demands that we require to integrate the blockchain technology with the industrial IoT applications. Since the blockchain helps in providing the trusted sharing services where the reliable information and data can be retrieved, the data (information) can be then traceable. At the same time, the blockchain is also immutable; thus it enhances the security as well. Therefore, integration of decentralized blockchain in IIoT system can enable better efficiency, transparency and guarantee security solutions.

5.2 Research contributions

The key contributions towards this work are mentioned below.

- We propose a novel "private blockchain-envisioned access control scheme for Pervasive Edge Computing (PEC) in IIoT environment, called PBACS-PECIIoT". The purpose behind applying the the private blockchain is that the transactions and registration credentials of the entities related to IIoT are confidential and private.

- In the proposed PBACS-PECIIoT, registration credentials obtained by a smart device (SD_i) and the gateway node (GN_j) are fetched from the Blockchain during the access

control phase for authentication and key agreement purposes. Additionally, it is also worth to notice that the registration credentials stored in the blockchain center (BC) are fetched by an edge server for key management purpose with its gateway node.

- After collecting the information securely from the deployed IoT smart devices by their respective gateway node(s), the information is securely delivered to the edge servers by their associated gateway nodes in form of transactions. The edge servers are then responsible for building the blocks, verifying and adding them in the private blockchain with the help of the proposed voting-based "Practical Byzantine Fault Tolerance (PBFT)" algorithm. The local ledgers are maintained by the edge servers in the blockchain center.

- A detailed security analysis including the formal security verification has been conducted. It demonstrates that PBACS-PECIIoT is secure against a number of potential attacks against passive/active adversaries.

- The "real testbed experiments for various cryptographic primitives with the help of widely-accepted Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5] have been performed under both server and Raspberry PI 3 platforms. These testbed experiments measure the computational time for the primitives with respect to these platforms. Moreover, a detailed comparative analysis among PBACS-PECIIoT and other related existing schemes has been performed. It shows the effectiveness and robustness of PBACS-PECIIoT over other schemes.

- The proposed PBACS-PECIIoT is also implemented through blockchain simulation study in order to measure its performance as well as computational time.

5.3 System model

In this section, the network as well as threat models used in the proposed scheme (PBACS-PECIIoT) are discussed.

5.3.1 Network model

The network model used in the proposed PBACS-PECIIoT is shown in Figure 5.1. The model shows different types of IIoT applications, such as mobile, car, aerospace, and food manufacturing industry.

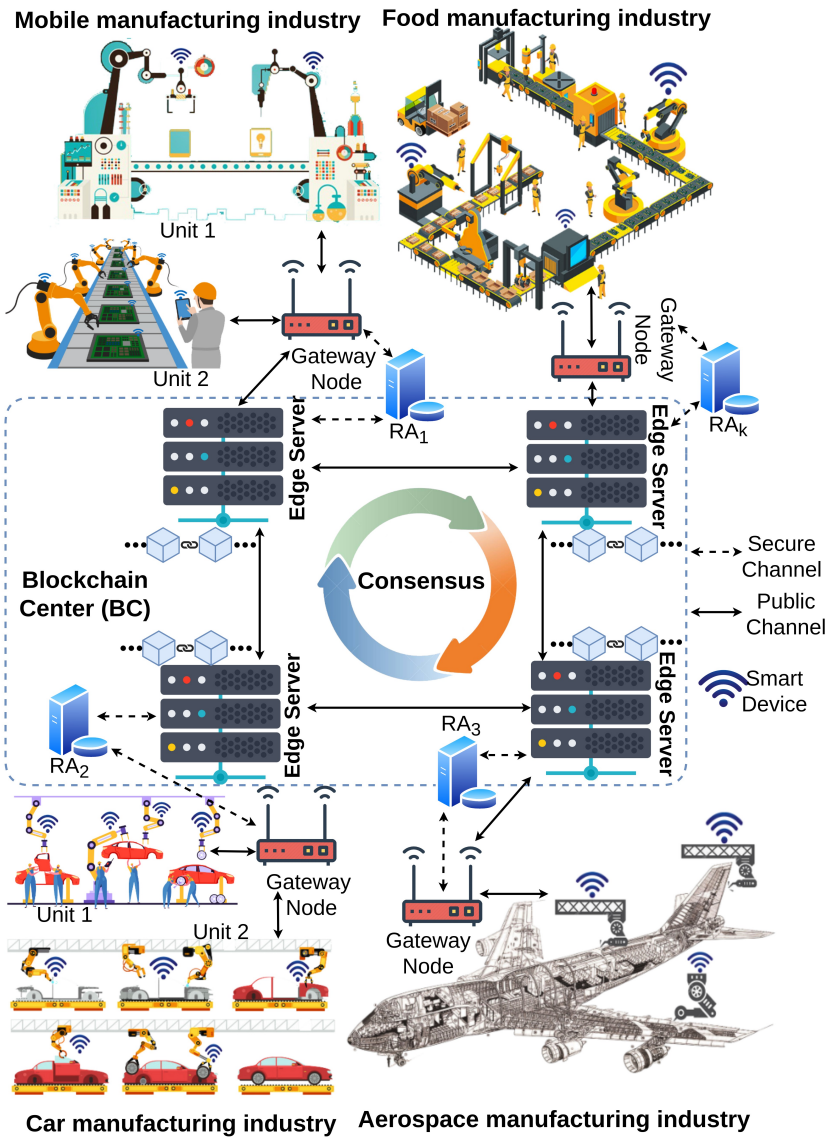

Figure 5.1: Blockchain-envisioned edge-based IIoT environment

Various smart IoT devices are attached with each unit of an industry, and all the smart devices, say $(SD_i|\ i = 1, 2, 3, \ldots, n_{sd})$ are connected with the associated gateway node(s), say $(GN_j|\ j = 1, 2, 3, \ldots, n_{gn})$. Each GN_j is connected with an edge server, say $(ES_l|l = 1, 2, 3, \ldots, n_{es})$. The registration of all the entities (SD_i, GN_j, ES_l) is executed by a trusted registration authority, say $(RA_k\ |k = 1, 2, 3, \ldots, n_{ra})$ for a particular application. Here, n_{sd}, n_{gn}, n_{es} and n_{ra} represent the number of IoT smart devices, gateway nodes, edge servers and RAs, respectively. All the registered ES_l form a P2P edge servers network, which is also called as the blockchain center. An edge server, being a leader node, say ES_l, runs a consensus

algorithm for creating, verifying, adding, and also mining the blocks in their *local ledgers* of the blockchain center.

5.3.2 Threat model

The following threat is model in analyzing the security of the proposed scheme in this chapter.

- The involved entities in an "IIoT environment" need to communicate over insecure channels. Therefore, an adversary $\mathcal{A}$ can take an opportunity to manipulate/compromise the data exchanged between them.
- In this work, we adapt the broadly-accepted "Dolev-Yao threat model (known as DY model)" [67]. Under the DY model, $\mathcal{A}$ "not only eavesdrops, but can also modify, delete and insert fake information during the communication among the entities".
- In addition, we also adapt the widely-known "Canetti and Krawczyk's model (CK-adversary model)" [37] which is presently a *de facto* threat model as compared to the DY model. Under the "CK-adversary model", $\mathcal{A}$ can compromise the "secret credentials shared between two communicating parties. This results the adversary $\mathcal{A}$ to compromise the past or future established session keys between the communicating parties by means of compromising the session states and session keys".
- We assume that end-point entities (IoT smart devices) are not trusted, whereas the gateway nodes and edge servers are semi-trusted and the registration authorities are fully trusted.
- Since it may not be possible to monitor the IoT smart devices in 24×7, a physical theft of some IoT smart devices by $\mathcal{A}$ may happen. It may then lead to compromise the secret credentials stored in the captured devices using some sophisticated "power analysis attacks" as mentioned in [141].

5.4 The proposed scheme: PBACS-PECIIoT

In this section, different phases relevant to the proposed private blockchain-envisioned access control scheme for edge-based IIoT environment, PBACS-PECIIoT, has been designed.

The proposed PBACS-PECIIoT has various phases, like *registration*, *access control*, *key management*, and *block creation, verification and addition in blockchain*. In Figure 5.2, we have

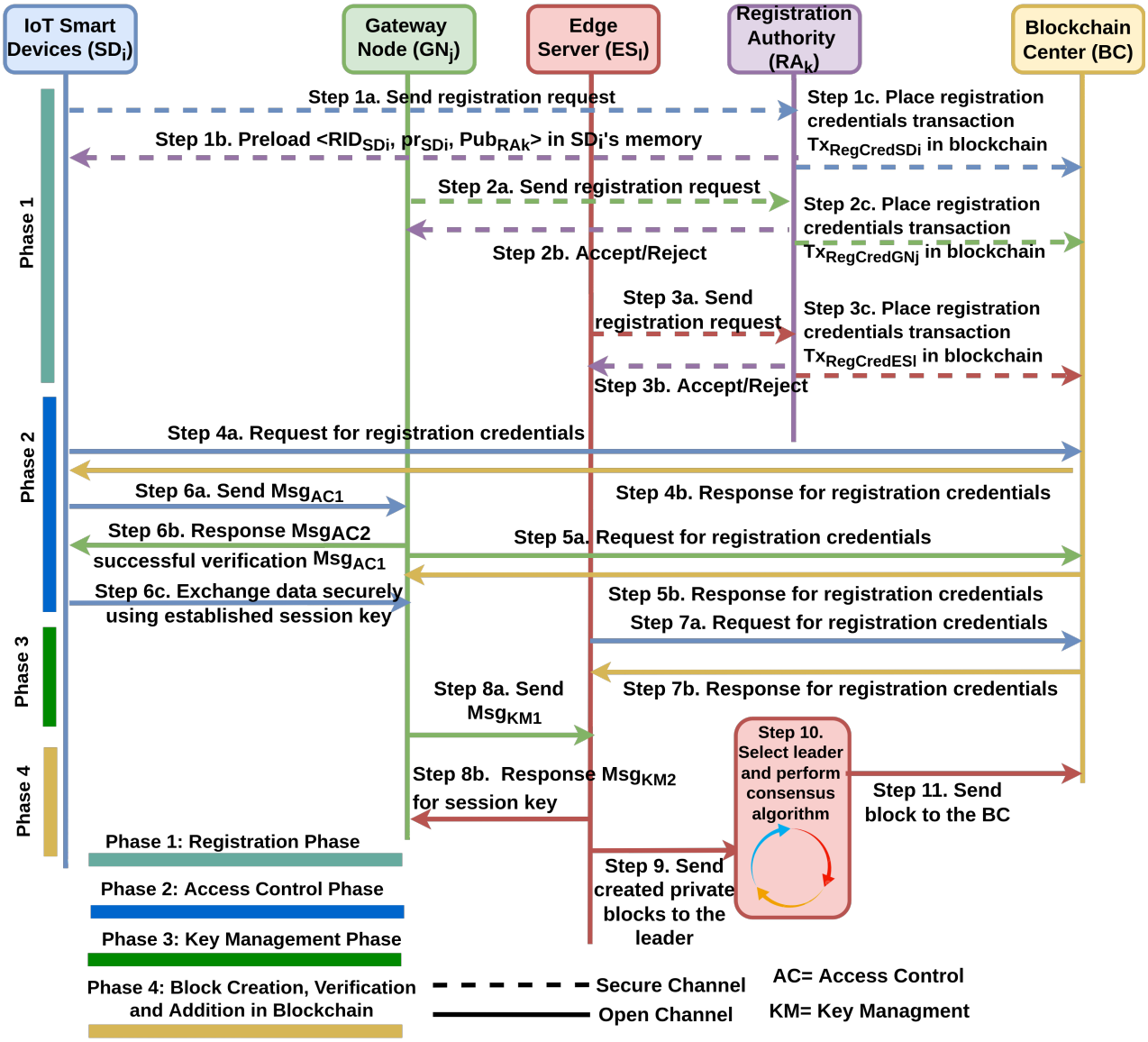

Figure 5.2: Illustration of complete process in PBACS-PECIIoT

illustrated the complete process in PBACS-PECIIoT. Note that in this work, we consider the access control that mainly consists of the following two tasks [94, 247]: 1) node authentication and 2) key agreement.

The idea behind the design of the proposed scheme is to mutually authenticate two communicating entities through the access control mechanism. It helps in establishing the secret session keys between the authorized entities in the IoT network so that they can secure communicate among each other for secure data delivery. In addition, the key management process

between a gateway node and its associate edge server helps in secure communication among them.

It is interesting to note that as compared to other existing schemes in the literature, in the proposed PBACS-PECIIoT, the registration credentials obtained by a smart device (SD_i) and the gateway node (GN_j) are fetched from the Blockchain during the access control phase for authentication and key agreement purposes. Additionally, the registration credentials stored in the blockchain center (BC) are also fetched by an edge server for key management purpose with its gateway node.

The deployed IoT smart devices first send the messages encrypted with their established session keys during the "access control phase" in Section 5.4.2 to their respective gateway node(s). The gateway nodes then send the information encrypted with their secret keys established during the"key management phase" in Section 5.4.3 to their respective edge servers. An in-charge edge server is responsible to create a block containing the encrypted transactions of information received from the gateway node(s) or IoT smart device(s) for a particular application.

Since PBACS-PECIIoT makes use of current system timestamps for safeguarding replay attacks, all the communicating entities, like "IoT smart devices", "gateway nodes" and "edge servers", are synchronized with their clocks. It is a widely accepted presumption applied in "different existing authentication and access control approaches under individual networking scenarios" [59, 224]. The list of symbols tabulated in Table 5.1 are utilized in describing and analyzing the proposed PBACS-PECIIoT.

5.4.1 Registration phase

During registration process of all the communicating entities, like "IoT smart devices", "gateway nodes" and "edge servers", each registration authority RA_k ($k = 1, 2, \ldots, n_{ra}$) selects the following system parameters. First of all, each RA_k will pick a large prime q and a "non-singular elliptic curve $E_q(\alpha, \beta) : y^2 = x^3 + \alpha x + \beta \pmod q$ over a finite (Galois) field $GF(q)$ with two constants $\alpha, \beta \in Z_q = \{0, 1, 2, \ldots, q-1\}$ such that $4\alpha^3 + 27\beta^2 \neq 0 \pmod q$ and the Elliptic Curve Discrete Logarithm Problem (ECDLP) becomes intractable due to sufficiently chosen large prime q". For instance, to make ECDLP intractable, q should be chosen at least 160 bits such that 160-bit "Elliptic Curve Cryptography (ECC)" security remains same as that for an 1024-bit RSA public key cryptosystem [22]. In addition, each RA_k also selects a base point G_k corresponding to the chosen elliptic curve $E_q(\alpha, \beta)$ whose order will be as big as q, and a common collision resistant cryptographic hash function $h(\cdot)$ (for example, we

Table 5.1: Notations and their meanings

Symbol	Meaning
RA	Trusted registration authority
SD_i, RID_{SD_i}	i^{th} IoT smart device and its pseudo identity
GN_j, RID_{GN_j}	j^{th} gateway node and its pseudo identity
ES_l, RID_{ES_l}	l^{th} edge server and its pseudo identity
$n_{sd}, n_{ra}, n_{gn}, n_{es}$	Number of IoT smart devices, RAs, gateway nodes and edge servers, respectively
$h(\cdot)$	A collision-resistant cryptographic one-way hash function
q	A large prime number
$GF(q)$	Galois finite field over prime q
$E_q(\alpha, \beta)$	A non-singular elliptic curve: $y^2 = x^3 + \alpha x + \beta \pmod q$ over $GF(q)$ with $\alpha, \beta \in Z_q = \{0, 1, \cdots, q-1\}$
G_k	A base point in $E_q(\alpha, \beta)$
$P + Q$	Elliptic curve point addition of two points $P, Q \in E_q(\alpha, \beta)$,
$u \cdot G_k$	Elliptic curve point multiplication; $u \cdot G_k = G_k + G_k + \cdots G_k$ (u times), $G_k \in E_q(\alpha, \beta)$, $u \in Z_q^*$
$u * v$	Modular multiplication of elements $u, v \in Z_q$
(pr_X, Pub_X)	Signature private-public key pair of an entity X, where $Pub_X = pr_X \cdot G_k$
$E(\cdot)/D(\cdot)$	ECC encryption/decryption algorithm
TS_x	Current system timestamp generated by an entity X
ΔT	Maximum transmission delay associated with a message
$\|, \oplus$	Data concatenation and exclusive-OR operators, respectively

can apply the Secure Hash Algorithm (SHA-256) hash function). Furthermore, each RA_k picks its own secret (private) master key mk_{RA_k} which is kept secret to itself, computes the respective public key $Pub_{RA_k} = mk_{RA_k} \cdot G$ and makes the system parameters $\{E_q(\alpha, \beta)$, G_k, $h(\cdot)$, $Pub_{RA_k}\}$ as public.

1) IoT smart devices registration

This phase occurs in offline mode prior to deployment of the IoT smart devices in their respective deployment areas by the associated registration authority RA_k. Note that the private and public key pairs for the IoT smart devices are generated by the registration authority RA_k, and these are pre-installed in IoT devices' memory before placing them in the network.

For registering each deployed IoT smart device SD_i for a particular application of IIoT, the respective registration authority RA_k first selects a unique identity ID_{SD_i} and then computes the corresponding pseudo-identity $RID_{SD_i} = h(ID_{SD_i} || mk_{RA_k})$ and temporal credential $TC_{SD_i} = h(RID_{SD_i} || mk_{RA_k} || RTS_{SD_i} || pr_{SD_i})$ of SD_i, where the private key of each SD_i is a random secret $pr_{SD_i} \in Z_q^* = \{1, 2, \ldots, q-1\}$ and its public key is $Pub_{SD_i} = pr_{SD_i} \cdot G_k$,

and RTS_{SD_i} is the registration timestamp of SD_i. The RA_k stores the information $\{RID_{SD_i}, pr_{SD_i}, Pub_{RA_k}\}$ into its secure memory prior to placement in IIoT application, and makes Pub_{SD_i} as public.

After that, the RA sends the registration related credentials $RegCred_{SD_i} = \{RID_{SD_i}, TC_{SD_i}, Pub_{SD_i}, E_q(\alpha, \beta), G_k\}$ to the blockchain center (BC) in the form of a transaction, say $Tx_{RegCred_{SD_i}} = \langle RID_{SD_i}, E_{Pub_{SD_i}}[RegCred_{SD_i}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{SD_i}]\rangle$, where $E_{Pub_X}(\cdot)$ and $D_{pr_X}(\cdot)$ represent the "ECC-based encryption and decryption using the public key Pub_X and private key pr_X of an entity," respectively, and $ECDSA.Sign(\cdot)$ and $ECDSA.Ver(\cdot)$ denote the "elliptic curve digital signature algorithm (ECDSA)-based signature generation and verification methods," respectively.

RA_k deletes RID_{SD_i}, TC_{SD_i} and (pr_{SD_i}, Pub_{SD_i}) from its database for "security reasons in order to avoid privileged-insider attack". It is worth noticing that the registration credentials need not to be sent back to the IoT devices after the registration process, because the credentials are directly stored in the memory of the deployed smart devices as well as in the blockchain center prior to their placement in the network.

2) Gateway nodes registration

To register a gateway node GN_j belonging to a particular IIoT application, its respective RA_k first picks the unique identity ID_{GN_j} and then computes its pseudo-identity $RID_{GN_j} = h(ID_{GN_j} \,||mk_{RA_k})$ and temporal credential $TC_{GN_j} = h(ID_{GN_j} \,||RTS_{GN_j} \,||mk_{RA_k})$ for GN_j, where RTS_{GN_j} is GN_j's registration time. Next, RA_k picks a "t-degree symmetric bivariate polynomial over the finite (Galois) field $GF(q)$ as $g_j(x, y) = \sum_{u=0}^{t} \sum_{v=0}^{t} a_{u,v} x^u y^v$, where the co-efficients $a_{u,v}$ are from Z_q, $g_j(x, y) = g_j(y, x)$, and $t >> n_{gn}$ and $t >> n_{es}$", as in Blundo *et al.*'s scheme [33]. Furthermore, RA_k calculates the polynomial share $g_j(RID_{GN_j}, y) = \sum_{u=0}^{t} \sum_{v=0}^{t} a_{u,v} (RID_{GN_j})^u y^v$ over $GF(q)$ and sends the registration credential $\{RID_{GN_j}\}$ to GN_j via a secure channel (for example, in person).

After receiving the registration credentials from RA_k, GN_j picks its own random secret (private) key $pr_{GN_j} \in Z_q^*$, computes the respective public key $Pub_{GN_j} = pr_{GN_j} \cdot G_k$, stores the private key pr_{GN_j} in its secure database and publishes the public key Pub_{GN_j}. Next, the RA_k sends the registration credentials $RegCred_{GN_j} = \{RID_{GN_j}, TC_{GN_j}, g_j(RID_{GN_j}, y), E_q(\alpha, \beta), G_k, \{(RID_{SD_i}, Pub_{SD_i})\}, Pub_{GN_j}\}$ to the blockchain center (BC) in the form of a transaction, say $Tx_{RegCred_{GN_j}} = \langle RID_{GN_j}, E_{Pub_{GN_j}}[RegCred_{GN_j}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{GN_j}]\rangle$. Note that only for the IoT smart devices SD_i that are associated with GN_j in a particular IIoT application, $\{(RID_{SD_i}, Pub_{SD_i})\}$ are available to GN_j. RA_k also deletes RID_{GN_j} and

TC_{GN_j} from its database for security reasons in order to avoid "privileged-insider attack", and stores Pub_{GN_j} in each associated SD_i's memory. Finally, GN_j stores the credentials $\{RID_{GN_j}, pr_{GN_j}\}$ in its secure database.

3) Edge servers registration

To register an edge server ES_l belonging to one or more GN_j for a particular IIoT application, the in-charge RA_k picks a unique identity ID_{ES_l} for ES_l and computes the pseudo-identity $RID_{ES_l} = h(mk_{RA_k} \,||ID_{ES_l})$ and the polynomial share $g_j(RID_{ES_l}, y) = \sum_{u=0}^{t} \sum_{v=0}^{t} a_{u,v}(RID_{ES_l})^u y^v$ over $GF(q)$. It is worth noticing that $g_j(x, y)$ is the common polynomial shared between GN_j and ES_l. After that RA_k sends the registration credentials $\{RID_{ES_l}\}$ to ES_l via secure channel.

Once the registration credentials are received by ES_l from RA_k, ES_l picks its own random private key $pr_{ES_l} \in Z_q^*$, calculates the corresponding public key $Pub_{ES_l} = pr_{ES_l} \cdot G_k$, and publishes Pub_{ES_l} as public. Here, the polynomial $g_j(x, y)$ is used to setup a symmetric secret key between the "gateway node GN_j" and its associated "edge server ES_l", which is further utilized in establishing the session key SK_{ES_l,GN_j} (SK_{GN_j,ES_l}) between them for secret communications (see Section 5.4.3).

The RA sends the registration credentials $RegCred_{ES_l} = \{RID_{ES_l}, g_j(RID_{ES_l}, y), TC_{ES_l}, Pub_{ES_l}, E_q(\alpha, \beta), G_k\}$ to the blockchain center (BC) in the form of a transaction, say $Tx_{RegCred_{ES_l}} = \langle RID_{ES_l}, E_{Pub_{ES_l}}[RegCred_{ES_l}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{ES_l}]\rangle$. Finally, ES_l needs to store the credentials $\{RID_{ES_l}, pr_{ES_l}\}$ in its secure database.

Remark 5.1. *It is noted that the IoT smart devices registration phase occurs in offline mode prior to deployment of the IoT devices in their respective deployment areas by the associated RA_k. In addition, the registration of the gateway nodes GN_j and the edge servers ES_l are executed in secure channels by the RA_k. As a result, there is no possibility of the impersonation attacks by an adversary (including the insider attacker) at the device/gateway/edge server registration phases because the encrypted registration credentials along with their signatures are placed into the blockchain.*

5.4.2 Access control phase

It is done between a "registered IoT smart device (SD_i)" and its respective "gateway node (GN_j)" for a particular application in IIoT. This phase helps to perform a "mutual authentication and session key establishment between SD_i and GN_j". Before initiating the access

control process, the SD_i and GN_j need to obtain the registration credentials that are already stored in the blockchain center (BC). Note that this is executed only once because the registration credentials obtained from the BC can be stored in the secure databases of both SD_i and GN_j. For this purpose, the following steps are involved:

i) *Registration credentials obtained by smart device (SD_i):* SD_i first sends a request message $RegCredReq_{SD_i} = \{RID_{SD_i}\}$ to the BC over open channel for obtaining its registration credentials. After receiving the request, the BC checks RID_{SD_i} and fetches the transaction $Tx_{RegCred_{SD_i}} = \langle RID_{SD_i}, E_{Pub_{SD_i}}[RegCred_{SD_i}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{SD_i}]\rangle$ and sends it to SD_i over public channel. SD_i upon receiving $Tx_{RegCred_{SD_i}}$, decrypts $E_{Pub_{SD_i}}[RegCred_{SD_i}]$ using the public key Pub_{RA_k} of the associated RA_k to extract $RegCred_{SD_i} = \{RID_{SD_i}, TC_{SD_i}, Pub_{SD_i}, E_q(\alpha, \beta), G_k\}$. Now, if the decrypted RID_{SD_i} matches with its received version, SD_i further validates the signature $ECDSA.Sign_{mk_{RA_k}}[RegCred_{SD_i}]$ by applying the $ECDSA.Ver(\cdot)$ algorithm using the RA_k's public key Pub_{RA_k}. If the signature is valid, SD_i then only stores $RegCred_{SD_i}$ in its memory for the mutual authentication and key establishment purpose.

ii) *Registration credentials obtained by gateway node (GN_j):* GN_j also sends a request message $RegCredReq_{GN_j} = \{RID_{GN_j}\}$ for obtaining the registration credentials to the BC over open channel. Once the request is processed by the BC, the BC checks RID_{GN_j}, fetches the transaction $Tx_{RegCred_{GN_j}} = \langle RID_{GN_j}, E_{Pub_{GN_j}}[RegCred_{GN_j}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{GN_j}]\rangle$ corresponding to RID_{GN_j} and sends it to GN_j over public channel. Moreover, GN_j upon receiving $Tx_{RegCred_{GN_j}}$, decrypts $E_{Pub_{GN_j}}[RegCred_{GN_j}]$ using the public key Pub_{RA_k} of the associated RA_k to extract $RegCred_{SD_i} = \{RID_{SD_i}, TC_{SD_i}, Pub_{SD_i}, E_q(\alpha, \beta), G_k\}$. On successful matching of the decrypted RID_{GN_j} with its received version, GN_j checks the validity of the signature $ECDSA.Sign_{mk_{RA_k}}[RegCred_{GN_j}]$ using the public key Pub_{RA_k}. Upon successful signature validation, GN_j stores $RegCred_{GN_j}$ in its secure database for the mutual authentication and key establishment purpose.

We now discuss the following steps needed for the access control between SD_i and GN_j with the help of the obtained registration credentials $RegCred_{SD_i}$ and $RegCred_{GN_j}$ from the BC, respectively.

- *Step AC1.* SD_i picks a random secret $rs_{SD_i} \in Z_q^*$ and the current timestamp TS_{SD_i} for computing $RS_{SD_i} = h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS_{SD_i}) \cdot G_k$. Furthermore, SD_i

computes signature on rs_{SD_i} as $Sig_{SD_i} = h(TC_{SD_i} \,||rs_{SD_i} \,||pr_{SD_i} \,||RID_{SD_i} \,||TS_{SD_i})$ $+h(RID_{SD_i} \,||Pub_{SD_i}||Pub_{GN_j}||TS_{SD_i}) * pr_{SD_i} \pmod q$. SD_i then sends access control request message $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS_{SD_i}\}$ to its corresponding gateway node GN_j via open channel.

- *Step AC2.* If the message Msg_{AC1} is received at time $TS^*_{SD_i}$, the GN_j first checks timeliness of received TS_{SD_i} by verifying $|TS^*_{SD_i} - TS_{SD_i}| < \Delta T$, where ΔT signifies the "maximum transmission delay with a message". If it is valid, GN_j retrieves Pub_{SD_i} corresponding to RID_{SD_i} from its own database and verifies the received signature Sig_{SD_i} by the condition: $Sig_{SD_i} \cdot G_k = RS_{SD_i} + h(RID_{SD_i} \,||Pub_{SD_i}||Pub_{GN_j}||TS_{SD_i}) \cdot Pub_{SD_i}$. Upon successful signature validation, GN_j validates SD_i as authentic device, creates a random secret $rs_{GN_j} \in Z^*_q$ and the current timestamp TS_{GN_j} for calculating $RS_{GN_j} = h(TC_{GN_j} \,||rs_{GN_j} \,||pr_{GN_j} \,||RID_{GN_j} \,||TS_{GN_j}) \cdot G_k$ and the Diffie-Hellman type key $DHK_{GN_j,SD_i} = h(TC_{GN_j} \,||rs_{GN_j} \,||pr_{GN_j} \,||RID_{GN_j} \,||TS_{GN_j}) \cdot RS_{SD_i}$. Furthermore, GN_j evaluates its own polynomial share $g_j(RID_{GN_j}, y)$ at the point $y = RID_{SD_i}$ to obtain $g_j(RID_{GN_j}, RID_{SD_i})$ and $y_{GN_j} = h(g_j(RID_{GN_j}, RID_{SD_i})$ $||Sig_{SD_i} \,||TS_{GN_j}) \oplus h(DHK_{GN_j,SD_i} \,||TS_{SD_i} \,||TS_{GN_j})$, and also computes the signature on rs_{GN_j} and DHK_{GN_j,SD_i} as $Sig_{GN_j} = h(TC_{GN_j} \,||rs_{GN_j} \,||pr_{GN_j} \,||RID_{GN_j} \,||TS_{GN_j})$ $+h(RID_{GN_j}||RID_{SD_i}||Pub_{GN_j}||DHK_{GN_j,SD_i} \,||y_{GN_j}) * pr_{GN_j} \pmod q$. Next, GN_j dispatches the access control response message $Msg_{AC2} = \{RID_{GN_j}, Sig_{GN_j}, RS_{GN_j}, y_{GN_j}, TS_{GN_j}\}$ to its corresponding SD_i via open channel.

- *Step AC3.* Let SD_i receive the message Msg_{AC2} at time $TS^*_{GN_j}$. SD_i then checks TS_{GN_j}'s validity by $|TS^*_{GN_j} - TS_{GN_j}| < \Delta T$ and if it is valid, SD_i fetches Pub_{GN_j} corresponding to RID_{GN_j} from its memory. SD_i calculates the Diffie-Hellman type key $DHK_{SD_i,GN_j} = h(TC_{SD_i} \,||rs_{SD_i} \,||pr_{SD_i} \,||RID_{SD_i} \,||TS_{SD_i}) \cdot RS_{GN_j}$ and $z_{SD_i} = y_{GN_j} \oplus$ $h(DHK_{SD_i,GN_j} \,||TS_{SD_i} \,||TS_{GN_j})$, which should be equal to $h(g_j(RID_{GN_j}, RID_{SD_i})$ $||Sig_{SD_i} \,||TS_{GN_j})$, and verifies the signature by the condition: $Sig_{GN_j} \cdot G_k = RS_{GN_j}$ $+h(RID_{GN_j} \,||RID_{SD_i} \,||Pub_{GN_j} \,||DHK_{SD_i,GN_j} \,||y_{GN_j}) \cdot Pub_{GN_j}$. Upon successful signature validation, SD_i authenticates GN_j as valid, generates current timestamp TS'_{SD_i}, and computes the session key shared with GN_j as $SK_{SD_i,GN_j} = h(DHK_{SD_i,GN_j} \,||z_{SD_i})$ and its verifier $SKV_{SD_i,GN_j} = h(SK_{SD_i,GN_j} \,||TS'_{SD_i})$. Finally, SD_i sends the acknowledgment message $Msg_{AC3} = \{SKV_{SD_i,GN_j}, TS'_{SD_i}\}$ to GN_j via public channel.

- *Step AC4.* Upon reception of the message Msg_{AC3} at time $TS^{**}_{SD_i}$, GN_j checks timeliness of received TS'_{SD_i} by verifying $|TS^{**}_{SD_i} - TS'_{SD_i}| < \Delta T$. If the validation passes,

Access Control Phase	
IoT smart device (SD_i)	Gateway node (GN_j)
Obtain registration credentials from the blockchain center (BC).	Obtain registration credentials from the blockchain center (BC).
Generates random secret $rs_{SD_i} \in Z_q^*$, current timestamp TS_{SD_i}.	
Calculate $RS_{SD_i} = h(TC_{SD_i} \|\|rs_{SD_i} \|\|pr_{SD_i} \|\|RID_{SD_i} \|\|TS_{SD_i}) \cdot G_k$, $Sig_{SD_i} = h(TC_{SD_i} \|\|rs_{SD_i} \|\|pr_{SD_i} \|\|RID_{SD_i} \|\|TS_{SD_i}) + h(RID_{SD_i} \|\|Pub_{SD_i} \|\|Pub_{GN_j} \|\|TS_{SD_i}) * pr_{SD_i} \pmod q$. $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS_{SD_i}\} \longrightarrow$ (via open channel)	Check if $\|TS^*_{SD_i} - TS_{SD_i}\| < \Delta T$? If valid, retrieve Pub_{SD_i} corresponding to RID_{SD_i}. Verify signature Sig_{SD_i}. If valid, generate random secret $rs_{GN_j} \in Z_q^*$, current timestamp TS_{GN_j}. Calculate $RS_{GN_j} = h(TC_{GN_j} \|\|rs_{GN_j} \|\|pr_{GN_j} \|\|RID_{GN_j} \|\|TS_{GN_j}) \cdot G_k$, $DHK_{GN_j,SD_i} = h(TC_{GN_j} \|\|rs_{GN_j} \|\|pr_{GN_j} \|\|RID_{GN_j} \|\|TS_{GN_j}) \cdot RS_{SD_i}$, $g_j(RID_{GN_j}, RID_{SD_i})$, $y_{GN_j} = h(g_j(RID_{GN_j}, RID_{SD_i}) \|\|Sig_{SD_i} \|\|TS_{GN_j}) \oplus h(DHK_{GN_j,SD_i} \|\|TS_{SD_i} \|\|TS_{GN_j})$, $Sig_{GN_j} = h(TC_{GN_j} \|\|rs_{GN_j} \|\|pr_{GN_j} \|\|RID_{GN_j} \|\|TS_{GN_j}) + h(RID_{GN_j} \|\|RID_{SD_i} \|\|Pub_{GN_j} \|\|DHK_{GN_j,SD_i} \|\|y_{GN_j}) * pr_{GN_j} \pmod q$. $Msg_{AC2} = \{RID_{GN_j}, Sig_{GN_j}, RS_{GN_j}, y_{GN_j}, TS_{GN_j}\} \longleftarrow$ (via open channel)
Check if $\|TS^*_{GN_j} - TS_{GN_j}\| < \Delta T$? If valid, retrieve Pub_{GN_j} corresponding to RID_{GN_j}. Calculate $DHK_{SD_i,GN_j} = h(TC_{SD_i} \|\|rs_{SD_i} \|\|pr_{SD_i} \|\|RID_{SD_i} \|\|TS_{SD_i}) \cdot RS_{GN_j}$, $z_{SD_i} = y_{GN_j} \oplus h(DHK_{SD_i,GN_j} \|\|TS_{SD_i} \|\|TS_{GN_j})$. $Sig_{GN_j} \cdot G_k = RS_{GN_j} + h(RID_{GN_j} \|\|RID_{SD_i} \|\|Pub_{GN_j} \|\|DHK_{SD_i,GN_j} \|\|y_{GN_j}) \cdot Pub_{GN_j}$? If valid, generate current timestamp TS'_{SD_i}. Compute $SK_{SD_i,GN_j} = h(DHK_{SD_i,GN_j} \|\|z_{SD_i})$, $SKV_{SD_i,GN_j} = h(SK_{SD_i,GN_j} \|\|TS'_{SD_i})$. $Msg_{AC3} = \{SKV_{SD_i,GN_j}, TS'_{SD_i}\} \longrightarrow$ (via open channel)	Checks if $\|TS^{**}_{SD_i} - TS'_{SD_i}\| < \Delta T$? If valid, calculate $SK_{GN_j,SD_i} = h(DHK_{GN_j,SD_i} \|\|h(g_j(RID_{GN_j}, RID_{SD_i}) \|\| Sig_{SD_i} \|\|TS_{GN_j}))$, $SKV_{GN_j,SD_i} = h(SK_{GN_j,SD_i} \|\|TS'_{SD_i})$. Verify if $SKV_{GN_j,SD_i} = SKV_{SD_i,GN_j}$? If valid, session key is valid and SK_{GN_j,SD_i} $(= SK_{SD_i,GN_j})$.

Figure 5.3: Summary of access control phase

GN_j then calculates the session key shared with SD_i as $SK_{GN_j,SD_i} = h(DHK_{GN_j,SD_i} \|\|h(g_j(RID_{GN_j}, RID_{SD_i}) \|\|Sig_{SD_i} \|\|TS_{GN_j}))$ and its verifier $SKV_{GN_j,SD_i} = h(SK_{GN_j,SD_i}$

$||TS'_{SD_i})$. If the verification condition: $SKV_{GN_j,SD_i} = SKV_{SD_i,GN_j}$ holds good, both GN_j and SD_i store the same session key SK_{GN_j,SD_i} $(= SK_{SD_i,GN_j})$ for their secret communications.

The above discussed access control phase is explained briefly in Figure 5.3.

5.4.3 Key management phase

Before the key management process starts, an edge server (ES_l) needs to obtain its registration credentials from the BS like SD_i and GN_j as discussed in Section 5.4.2. Note that GN_j already obtained its registration credentials from the BC and stored in its secure database. ES_l issues a request $RegCredReq_{ES_l} = \{RID_{ES_l}\}$ for obtaining its registration credentials to the BC over open channel. Once the BC checks the validity of RID_{ES_l}, it fetches the corresponding transaction $Tx_{RegCred_{ES_l}} = \langle RID_{ES_l}, E_{Pub_{ES_l}}[RegCred_{ES_l}], ECDSA.Sign_{mk_{RA_k}}[RegCred_{ES_l}]\rangle$ and sends it to ES_l over public channel. After decrypting $E_{Pub_{ES_l}}[RegCred_{ES_l}]$ using the private key pr_{ES_l}, ES_l extracts $RegCred_{ES_l} = \{RID_{ES_l}, g_j(RID_{ES_l}, y), TC_{ES_l}, Pub_{ES_l}, E_q(\alpha, \beta), G_k\}$ and checks the validity of both the decrypted RID_{ES_l} and the signature $ECDSA.Sign_{mk_{RA_k}}[RegCred_{ES_l}]$. If all these are valid, ES_l stores $RegCred_{ES_l}$ in its secure database which is used for the key management purpose as discussed below.

The sole goal of this phase is to setup a (pairwise) secret key between a gateway node (GN_j) and its corresponding edge server (ES_l) for their communications. This phase involves the exchange of three messages, namely Msg_{KM1}, Msg_{KM2} and Msg_{KM3} between GN_j and ES_l, that use the registration credentials obtained from the BC along with random generated secrets and current timestamps. After verifying the message Msg_{KM1}, ES_l generates the session key shared with GN_j and sends the message Msg_{KM2} to GN_j. Validation of Msg_{KM2} by GN_j assures mutual authentication between GN_j and ES_l. Furthermore, verification of Msg_{KM3} guarantees that the "session key established between GN_j and ES_l are same and legitimate".

We now explain the followings stages:

- *Step KM1.* The initiator GN_j first creates a random secret $rs_{GN_{j1}} \in Z_q^*$ and a current timestamp $TS_{GN_{j1}}$ in order to calculate $RS_{GNj1} = h(RID_{GN_j} ||rs_{GN_{j1}} ||TC_{GN_j} ||TS_{GN_{j1}} ||pr_{GN_j}) \cdot G_k$. Next, GN_j calculates the signature on $rs_{GN_{j1}}$ as $Sig_{GN_{j1}} = h(RID_{GN_j} ||rs_{GN_{j1}} ||TC_{GN_j} ||TS_{GN_{j1}} ||pr_{GN_j}) + h(RID_{GN_j} ||Pub_{GN_j} ||Pub_{ES_l} ||TS_{GN_{j1}}) * pr_{GN_j}$

(mod q) and dispatches the request message $Msg_{KM1} = \{RID_{GN_j}, RS_{GN_{j1}}, Sig_{GN_{j1}}, TS_{GN_{j1}}\}$ to its respective ES_l via open channel.

- *Step KM2.* After receiving Msg_{KM1}, if the timeliness check of the received timestamp $TS_{GN_{j1}}$ passes, ES_l proceeds to verify signature $Sig_{GN_{j1}}$ by the condition: $Sig_{GN_{j1}} \cdot G_k = RS_{GN_{j1}} + h(RID_{GN_j} || Pub_{GN_j} || Pub_{ES_l} || TS_{GN_{j1}}) \cdot Pub_{GN_j}$. Now, if the signature is valid, ES_l treats GN_j as valid, and creates a random secret $rs_{ES_{l2}} \in Z_q^*$ and current timestamp $TS_{ES_{l2}}$ to calculate $RS_{ES_{l2}} = h(RID_{ES_l} || pr_{ES_l} || rs_{ES_{l2}} || TS_{ES_{l2}}) \cdot G_k$ and the Diffie-Hellman type key $DHK_{ES_l,GN_j} = h(RID_{ES_l} || pr_{ES_l} || rs_{ES_{l2}} || TS_{ES_{l2}}) \cdot RS_{GN_{j1}}$. After these computations, ES_l also evaluates its own polynomial share $g_j(RID_{ES_l}, y)$ at the point $y = RID_{GN_j}$ to have the secret $g_j(RID_{ES_l}, RID_{GN_j})$, and then computes the secret pairwise key shared with GN_j as $SK_{ES_l,GN_j} = h(DHK_{ES_l,GN_j} || g_j(RID_{ES_l}, RID_{GN_j}))$ and a signature on both $rs_{ES_{l2}}$ and SK_{ES_l,GN_j} as $Sig_{ES_{l2}} = h(RID_{ES_l} || pr_{ES_l} || rs_{ES_{l2}} || TS_{ES_{l2}}) + h(Pub_{ES_l} || SK_{ES_l,GN_j} || TS_{ES_{l2}}) * pr_{ES_l}$ (mod q). Next, ES_l dispatches the response message $Msg_{KM2} = \{RID_{ES_l}, RS_{ES_{l2}}, Sig_{ES_{l2}}, TS_{ES_{l2}}\}$ to GN_j via open channel.

- *Step KM3.* After checking the timeliness of the timestamp $TS_{ES_{l2}}$ in the received message Msg_{KM2}, GN_j computes the "Diffie-Hellman type key" $DHK_{GN_j,ES_l} = h(RID_{GN_j} || rs_{GN_{j1}} || TC_{GN_j} || TS_{GN_{j1}} || pr_{GN_j}) \cdot RS_{ES_{l2}}$, $g_j(RID_{GN_j}, RID_{ES_l}) = g_j(RID_{ES_l}, RID_{GN_j})$ using its own polynomial share $g_j(RID_{GN_j}, y)$ as $g_j(x, y) = g_j(y, x)$ and the session key shared with ES_l as $SK_{GN_j,ES_l} = h(DHK_{GN_j,ES_l} || g_j(RID_{GN_j}, RID_{ES_l}))$. Next, GN_j verifies the signature $Sig_{ES_{l2}}$ as $Sig_{ES_{l2}} \cdot G_k = RS_{ES_{l2}} + h(Pub_{ES_l} || SK_{GN_j,ES_l} || TS_{ES_{l2}}) \cdot Pub_{ES_l}$. If the signature validation passes, GN_j also treats ES_l as authentic entity.

Finally, both GN_j and ES_l require to store the same secret pairwise key SK_{GN_j,ES_l} ($= SK_{ES_l,GN_j}$) for their secure communications. This key management phase is briefly explained in Figure 5.4.

5.4.4 Block creation, verification and addition in blockchain

In this section, we elaborate the process of block creation, verification and addition of that block in the blockchain. For this issue, the IoT smart devices first send the messages encrypted with their established session keys as described during the "access control phase" in Section 5.4.2 to their respective gateway node(s). In turn, the gateway nodes also send the information

Key Management Phase	
Gateway node (GN_j)	Edge server (ES_l)
	Obtain registration credentials from the blockchain center (BC).
Create random secret $rs_{GN_{j1}} \in Z_q^*$, current timestamp $TS_{GN_{j1}}$. Calculate $RS_{GNj1} = h(RID_{GN_j} \|\|rs_{GN_{j1}} \|\|TC_{GN_j} \|\|TS_{GN_{j1}} \|\|pr_{GN_j}) \cdot G_k$, $Sig_{GN_{j1}} = h(RID_{GN_j} \|\|rs_{GN_{j1}} \|\|TC_{GN_j} \|\|TS_{GN_{j1}} \|\|pr_{GN_j}) + h(RID_{GN_j} \|\|Pub_{GN_j} \|\|Pub_{ES_l} \|\|TS_{GN_{j1}}) * pr_{GN_j} \pmod q$. $Msg_{KM1} = \{RID_{GN_j}, RS_{GN_{j1}}, Sig_{GN_{j1}}, TS_{GN_{j1}}\}$ (via open channel)	Check validity of timestamp $TS_{GN_{j1}}$. If valid, further verify signature $Sig_{GN_{j1}}$. If valid, create random $rs_{ES_{l2}} \in Z_q^*$, current timestamp $TS_{ES_{l2}}$. Calculate $RS_{ES_{l2}} = h(RID_{ES_l} \|\|pr_{ES_l} \|\|rs_{ES_{l2}} \|\|TS_{ES_{l2}}) \cdot G_k$, $DHK_{ES_l,GN_j} = h(RID_{ES_l} \|\|pr_{ES_l} \|\|rs_{ES_{l2}} \|\|TS_{ES_{l2}}) \cdot RS_{GN_{j1}}$.
Check validity of $TS_{ES_{l2}}$. If valid, compute $DHK_{GN_j,ES_l} = h(RID_{GN_j} \|\|rs_{GN_{j1}} \|\|TC_{GN_j} \|\|TS_{GN_{j1}} \|\|pr_{GN_j}) \cdot RS_{ES_{l2}}$ $SK_{GN_j,ES_l} = h(DHK_{GN_j,ES_l} \|\| g_j(RID_{GN_j}, RID_{ES_l}))$. Verify signature $Sig_{ES_{l2}}$. If signature is valid, established key is legitimate, and SK_{GN_j,ES_l} $(= SK_{ES_l,GN_j})$.	Obtain $g_j(RID_{ES_l}, RID_{GN_j})$, and compute $SK_{ES_l,GN_j} = h(DHK_{ES_l,GN_j} \|\| g_j(RID_{ES_l}, RID_{GN_j}))$, and signature $Sig_{ES_{l2}} = h(RID_{ES_l} \|\|pr_{ES_l} \|\|rs_{ES_{l2}} \|\|TS_{ES_{l2}}) + h(Pub_{ES_l} \|\|SK_{ES_l,GN_j} \|\|TS_{ES_{l2}}) * pr_{ES_l} \pmod q$. $Msg_{KM2} = \{RID_{ES_l}, RS_{ES_{l2}}, Sig_{ES_{l2}}, TS_{ES_{l2}}\}$ (via open channel)

Figure 5.4: Summary of key management phase

encrypted with their secret keys established during the"key management phase" in Section 5.4.3 to their respective edge servers. An edge server ES_l is then responsible to construct a block containing the encrypted transactions of information received from the gateway node(s) or IoT smart device(s) for a particular application. Here, the ECC public key Pub_{ES_l} is used for generating the encrypted transactions because the information is strictly private and confidential with respect to an IIoT application. ES_l creates Merkle tree root on the encrypted transactions along with timestamp and random number. The current hash block is computed as $CurBH = h(BlkV \|\|PreBH \|\|MrkTR \|\|IdtT_m \|\|TS_{ES_l} \|\|CreB_{ID} \|\|Pub_{ES_l} \|\|\{E_{Pub_{ES_l}}(Tx_i) | i = 1, 2, \cdots, t_n\})$ and the signature on $CurBH$ as $Sig_{Block_k} = ECDSA.Sig_{pr_{ES_l}}(CurBH)$ where $ECDSA.Sig(\cdot)$ denotes the "ECDSA signature generation algorithm". The overall

Block Header	
Block Version	$BlkV$
Previous Block Hash	$PreBH$
Merkle Tree Root	$MrkTR$
Industry Type	$IT_m, m = 1, 2, \cdots, w$ (Type of an IIoT applications)
Timestamp	TS_{ES_l}
Creator of Block	$CreB_{ID}$ (Identity of an ES_l in P2P network)
Public Key of Signer ES_l	Pub_{ES_l}
Block Payload (Encrypted Transactions)	
List of t_n Encrypted Transactions $\#i$ (Tx_i)	$\{E_{Pub_{ES_l}}(Tx_i) \mid i = 1, 2, \cdots, t_n\}$
Current Block Hash	$CurBH$
Signature on $CurBH$	$Sig_{Block_k} = ECDSA.Sig_{pr_{ES_l}}(CurBH)$

Figure 5.5: Architecture of a block $Block_k$ for various transactions

structure of a block $Block_k$ is shown in Figure 5.5. The encryption is used in the transactions to make the transactions private with the edge server so that other P2P servers can not decrypt without private key of the particular edge server. Since the encryption is performed with the help of Pub_{ESl}, so only particular edge server associated with an application can see and decrypt the data. Finally, through the consensus algorithm provided in Algorithm 3, a leader among the group edge servers in the P2P network is selected using the existing leader selection algorithm [239] and then the leader sends the created block, say $Block_k$ to its peer nodes to have the consensus among them for verifying and adding the block in their local ledgers of blockchain center containing the fog servers. Note that we have applied the "Practical Byzantine Fault Tolerance (PBFT)" algorithm [38] for consensus purpose. However, we have provided the voting-based PBFT version as the proposed consensus algorithm, in which a leader L selected among the P2P network, generates a current timestamp TS_L and a random number r_L to perform voting process. L then creates signature Sig_L using the ECDSA signature generation algorithm with its own private key pr_L on the message $h(Block_k \,||TS_L\,||r_L\,||VTR_{req})$, where VTR_{req} is voting request, and sends the request message $\langle Block_k, Sig_L, E_{Pub_{ES_l}}[r_L, VTR_{req}], TS_L\rangle$ to each other edge server ES_l via public channel. After successful validation of timestamp TS_L, Merkle tree root $MrkTR$, current block hash $CurBH$ and signature on block Sig_{Block_k}, each other node in the P2P network sends the response message $\langle E_{Pub_L}[r_L, TS_{ES_l}, VTR_{res}], TS_{ES_l}\rangle$ to L via public channel, where VTR_{res} is the "voting response" and TS_{ES_l} is the "current timestamp".

An edge server associated with an IIoT application is responsible to create the blocks and store them into the blockchain after the consensus process as described in Algorithm 3.

Algorithm 3 Voting-based consensus for verification and addition of a block ($Block_k$)

Input: $Block_k$ = {Block Header, Block Payload, $CurBH$, Sig_{Block_k}}; private-public key pairs (pr_{ES_l}, Pub_{ES_l}), and $n_{f_{ES_l}}$ represents the number of faulty nodes in P2P network.

Output: Commitment & addition of $Block_k$ in the blockchain after successful validation.

1: Assume a leader has been selected by ES_l, say L using the leader selection algorithm [239].
2: L generates current timestamp TS_L and a random number r_L to perform voting process. L creates signature Sig_L using ECDSA signature generation algorithm with its own private key pr_L on the message $h(Block_k \,||TS_L\,||r_L\,||VTR_{req})$, where VTR_{req} is voting request.
3: L sends the request message $\langle Block_k, Sig_L, E_{Pub_{ES_l}}[r_L, VTR_{req}], TS_L\rangle$ to each other edge server ES_l via public channel.
4: After receiving request message, each ES_l checks timestamp TS_L and if it is valid, it computes $(r_L, VTR_{req}) = D_{pr_{ES_l}}[E_{Pub_{ES_l}}[r_L, VTR_{req}]]$, verifies Sig_L using ECDSA signature verification algorithm [104].
5: If the signature is valid, ES_l further verifies $MrkTR$, $CurBH$, and Sig_{Block_k} on received block $Block_k$.
6: After all the successful validations, ES_l sends the response message $\langle E_{Pub_L}[r_L, TS_{ES_l}, VTR_{res}], TS_{ES_l}\rangle$ to L via public channel, where VTR_{res} is the voting response and TS_{ES_l} is current timestamp.
7: Assume that $VCount$ represents the number of valid votes. Set $VCount \leftarrow 0$.
8: **for** each received message $\langle E_{Pub_L}[r_L, TS_{ES_l}, VTR_{res}], TS_{ES_l}\rangle$ from other edge nodes ES_l **do**
9: L first checks validity of timestamp TS_{ES_l}, decrypts message using its private key pr_L and validates r_L, TS_{ES_l} and VTR_{esp}. If all are valid, set $VCount = VCount + 1$.
10: **end for**
11: **if** $(VCount > 2n_{f_{ES_l}} + 1)$ **then**
12: L sends commit response for successful verification of $Block_k$ to its all followers ES_l.
13: Each ES_l and L then add $Block_k$ to their local ledgers.
14: **end if**

Remark 5.2. *Due to hostile environment/power exhaustion of IoT smart devices, the devices may be either physically captured or shut down. To continue the functionality of IIoT environment, new smart device, say SD^{new} needs to be added. Prior to deployment, SD^{new} is required to register by the trusted registration authority RA_k in that particular application*

where existing other smart devices are already there. For registering SD^{new}, RA_k needs to follow the same steps as described in the IoT smart devices registration (see Section 5.4.1).

Remark 5.3. *In this work, we have mainly considered the private blockchain scenario where the data is private and confidential with respect to each edge server. However, there are some applications, where the data needs to be shared inside the system. Thus, encrypting them will make the entities without the secret keys unable to decrypt the data and use the data. In this case, the edge servers can maintain a group (secret) key among them so that the selected shared data (transactions) can be now encrypted with the help of the group key using symmetric key encryption. Hence, the shared encrypted data can be decrypted by other edge servers using the same group key.*

5.5 Security analysis

In this section, we discourse the security analysis to show that the proposed PBACS-PECIIoT is resilient against the following potential attacks. In addition, we also show the correctness proofs of the session keys establishment during both the access control and key management phases.

5.5.1 Correctness proof

We show that the established secret keys among different communicating entities during the access control and key management phases are correct using the theorems 5.1 and 5.2.

Theorem 5.1. *The session keys established by an IoT smart device (SD_i) and its gateway node (GW_j) are the same during the access control phase provided in Section 5.4.2.*

Proof. During the access control phase discussed in Section 5.4.2, an IoT smart device, SD_i, computes the session key shared with its gateway node, GN_j, as $SK_{SD_i,GN_j} = h(DHK_{SD_i,GN_j} ||z_{SD_i})$. On the other side, the gateway node, GN_j, also computes the session key shared with SD_i as $SK_{GN_j,SD_i} = h(DHK_{GN_j,SD_i} ||h(g_j(RID_{GN_j}, RID_{SD_i}) ||Sig_{SD_i} ||TS_{GN_j}))$.

Now, we have, $z_{SD_i} = y_{GN_j} \oplus h(DHK_{SD_i,GN_j} ||TS_{SD_i} ||TS_{GN_j}) = h(g_j(RID_{GN_j}, RID_{SD_i}) ||Sig_{SD_i} ||TS_{GN_j})$. In addition, we have, and $DHK_{SD_i,GN_j} = h(TC_{SD_i} ||rs_{SD_i} ||pr_{SD_i} ||RID_{SD_i} ||TS_{SD_i}) \cdot RS_{GN_j}$, $DHK_{GN_j,SD_i} = h(TC_{GN_j} ||rs_{GN_j} ||pr_{GN_j} ||RID_{GN_j} ||TS_{GN_j}) \cdot RS_{SD_i}$, $RS_{SD_i} = h(TC_{SD_i} ||rs_{SD_i} ||pr_{SD_i} ||RID_{SD_i} ||TS_{SD_i}) \cdot G_k$, and $RS_{GN_j} = h(TC_{GN_j} ||rs_{GN_j} ||pr_{GN_j} ||RID_{GN_j} ||TS_{GN_j}) \cdot G_k$. Since $z_{SD_i} = h(g_j(RID_{GN_j}, RID_{SD_i}) ||Sig_{SD_i} ||TS_{GN_j})$, in

order to show $SK_{SD_i,GN_j} = SK_{GN_j,SD_i}$, it suffices to show that $DHK_{SD_i,GN_j} = DHK_{GN_j,SD_i}$. It then follows that

$$\begin{aligned} DHK_{SD_i,GN_j} &= h(TC_{SD_i}||rs_{SD_i}||pr_{SD_i}||RID_{SD_i}||TS_{SD_i}) \cdot RS_{GN_j} \\ &= [h(TC_{SD_i}||rs_{SD_i}||pr_{SD_i}||RID_{SD_i}||TS_{SD_i}) * \\ &\quad h(TC_{GN_j}||rs_{GN_j}||pr_{GN_j}||RID_{GN_j}||TS_{GN_j})] \cdot G_k \\ &= [h(TC_{GN_j}||rs_{GN_j}||pr_{GN_j}||RID_{GN_j}||TS_{GN_j}) * \\ &\quad h(TC_{SD_i}||rs_{SD_i}||pr_{SD_i}||RID_{SD_i}||TS_{SD_i})] \cdot G_k \\ &= h(TC_{GN_j}||rs_{GN_j}||pr_{GN_j}||RID_{GN_j}||TS_{GN_j}) \cdot RS_{SD_i} \\ &= DHK_{GN_j,SD_i}. \end{aligned}$$

Hence, $SK_{SD_i,GN_j} = SK_{GN_j,SD_i}$ and the theorem follows. □

Theorem 5.2. *The session keys established by a gateway node (GW_j) and its respective edge server ES_l are the same during the key management phase provided in Section 5.4.3.*

Proof. During the key management phase (see Section 5.4.3), ES_l computes the secret pairwise key shared with GN_j as $SK_{ES_l,GN_j} = h(DHK_{ES_l,GN_j}|| \ g_j(RID_{ES_l}, \ RID_{GN_j}))$, where $DHK_{ES_l,GN_j} = h(RID_{ES_l} \ ||pr_{ES_l} \ ||rs_{ES_{l2}} \ ||TS_{ES_{l2}}) \cdot RS_{GN_{j1}}$ and $RS_{GNj1} = h(RID_{GN_j} \ ||rs_{GN_{j1}} \ ||TC_{GN_j} \ ||TS_{GN_{j1}} \ ||pr_{GN_j}) \cdot G_k$. On the other side, GN_j also computes the session key shared with ES_l as $SK_{GN_j,ES_l} = h(DHK_{GN_j,ES_l}|| \ g_j(RID_{GN_j}, \ RID_{ES_l}))$, where $DHK_{GN_j,ES_l} = h(RID_{GN_j} \ ||rs_{GN_{j1}} \ ||TC_{GN_j} \ ||TS_{GN_{j1}} \ ||pr_{GN_j}) \cdot RS_{ES_{l2}}$ and $RS_{ES_{l2}} = h(RID_{ES_l} \ ||pr_{ES_l} \ ||rs_{ES_{l2}} \ ||TS_{ES_{l2}}) \cdot G_k$.

Since the bivariate polynomial $g_j(x, y)$ is symmetric, it follows that $g_j(RID_{GN_j}, \ RID_{ES_l}) = g_j(RID_{ES_l}, \ RID_{GN_j})$. To prove $SK_{ES_l,GN_j} = SK_{GN_j,ES_l}$ it is sufficient to show that $DHK_{ES_l,GN_j} = DHK_{GN_j,ES_l}$. It is worth noticing that

$$\begin{aligned} DHK_{ES_l,GN_j} &= h(RID_{ES_l}||pr_{ES_l}||rs_{ES_{l2}}||TS_{ES_{l2}}) \cdot RS_{GN_{j1}} \\ &= [h(RID_{ES_l}||pr_{ES_l}||rs_{ES_{l2}}||TS_{ES_{l2}}) * \\ &\quad h(RID_{GN_j}||rs_{GN_{j1}}||TC_{GN_j}||TS_{GN_{j1}}||pr_{GN_j})] \cdot G_k \\ &= [h(RID_{GN_j}||rs_{GN_{j1}}||TC_{GN_j}||TS_{GN_{j1}}||pr_{GN_j}) * \\ &\quad h(RID_{ES_l}||pr_{ES_l}||rs_{ES_{l2}}||TS_{ES_{l2}})] \cdot G_k \\ &= h(RID_{GN_j}||rs_{GN_{j1}}||TC_{GN_j}||TS_{GN_{j1}}||pr_{GN_j}) \cdot RS_{ES_{l2}} \\ &= DHK_{GN_j,ES_l}. \end{aligned}$$

Hence, it follows that $SK_{ES_l,GN_j} = SK_{GN_j,ES_l}$. □

5.5.2 Formal security analysis under ROR model

In this section, we discuss about the "session key security under broadly-recognized Real-Or-Random (ROR) oracle model [10] to show that PBACS-PECIIoT is secure against an adversary $\mathcal{A}$ for deriving the session-key between a smart device (SD_i) and a gateway node (GN_j) during the access control phase" described in Section 5.4.2. It is worth noticing that the "ROR-model based security analysis" provides the semi-formal security proof where the advantage of an adversary, say $\mathcal{A}$, is computed, and $\mathcal{A}$ attempts to derive the session key among two communicating entities in the network.

1) Random oracle model

We first describe the respective security model that is based on the works by Bellare *et al.* [23] and Wu *et al.* [224], for the proposed scheme, that goes through a sequence of the interactive games between a challenger and an adversary. Here, the main intention is to prove that the proposed scheme provides the session key security against the adversary.

The adversary $\mathcal{A}$ is permitted to execute the following queries for deriving the session key:

- $Execute(\Theta_{SD_i}^{a_1}, \Theta_{GN_j}^{a_2})$: $\mathcal{A}$ carries out this query to eavesdrop the messages exchanged between SD_i and GN_j.
- $CorruptSD(\Theta_{SD_i}^{a_1})$: It allows $\mathcal{A}$ to extract "the credentials stored in a stolen or lost SD_i's memory".
- $Reveal(\Theta^a)$: By executing this query, the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$ is exposed to $\mathcal{A}$ that is shared between Θ^a and its respective associate.
- $Test(\Theta^a)$: $\mathcal{A}$ is allowed to perform Θ^a to verify if the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$ is real or a random key.

Definition 5.1 of the semantic security is used to show the session key security of PBACS-PECIIoT in Theorem 5.3. In addition, as discussed in [42], a "collision-resistant one-way cryptographic hash function $h(\cdot)$ is accessed to all the involved participants including the adversary $\mathcal{A}$". As a result, we also model "$h(\cdot)$ as a random oracle, say $hash$".

The ROR model is associated with the following components:

- **Participants.** As we consider the access control between smart device SD_i and gateway node GN_j mentioned in Section 5.4.2, two participants, namely SD_i and GN_j are engaged for communication, and apart from these entities the registration authority RA_k is

also involved during offline registration purpose and dynamic node addition phase. The notations $\Theta_{SD_i}^{a_1}$ and $\Theta_{GN_j}^{a_2}$ signify the a_1^{th} and a_2^{th} instances of SD_i and GN_j, respectively. These instances are known as the "random oracles".

- **Accepted state.** An instance Θ^a will enter in its "accepted state" once it goes to an accept state when the last valid protocol message is received. If all the communicated messages (sent and received) are put in an ordered sequence, it creates a "session identification *sid* of Θ^a for the current session".

- **Partnering.** Two instances (Θ^{a_1} and Θ^{a_2}) will be the partners to each other if the following are fulfilled: a) Θ^{a_1} and Θ^{a_2} are in "accepted states"; b) Θ^{a_1} and Θ^{a_2} share the same *sid* and also "mutually authenticate each other"; and c) Θ^{a_1} and Θ^{a_2} are "mutual partners of each other".

- **Freshness.** An instance $\Theta_{SD_i}^{a_1}$ or $\Theta_{GN_j}^{a_2}$ is *fresh* if the established session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$ shared between SD_i and GN_j is not revealed to $\mathcal{A}$ using the Reveal(Θ^a) query described above.

We now define the "semantic security" in Definition 5.1 prior to prove Theorem 5.3.

Definition 5.1 (Semantic security)**.** *The "advantage of an adversary $\mathcal{A}$ running in polynomial time t in breaking the semantic security of the proposed PBACS-PECIIoT for deriving the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$ among a smart device SD_i and a gateway node GN_j" in a particular session during the access control phase (ACP) is $Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) = |2Pr[c' = c] - 1|$, where "$c$ and c' are respectively the correct and guessed bits".*

2) Provable security

In this section, we apply the random oracle model discussed above in order to prove that the proposed scheme provides the session key security that is described in Theorem 5.3.

Theorem 5.3. *The advantage $Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t)$ of an adversary $\mathcal{A}$ running in polynomial time t in order to derive the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$ established between SD_i and GN_j in a particular session during the access control phase (ACP) for the proposed PBACS-PECIIoT is $Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) \leq \frac{q_h^2}{|hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t)$, where q_h, $|hash|$, and $Adv_{\mathcal{A}}^{ECDDHP}(t)$ represent the "number of hash queries", the "range space of a one-way collision-resistant hash function $h(\cdot)$", and the "advantage of breaking the Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP)", respectively.*

Proof. A similar proof is followed here as in [42]. In the proposed PBACS-PECIIoT, we consider three games, namely $Game_i^{\mathcal{A}}$ for the adversary $\mathcal{A}$, $i = 0, 1, 2$. We define $Succ_{Game_i}^{\mathcal{A}}$ as an event wherein $\mathcal{A}$ can guess the random bit c correctly in the game $Game_i^{\mathcal{A}}$. Therefore, $\mathcal{A}$'s advantage to win the $Game_i^{\mathcal{A}}$ in the proposed PBACS-PECIIoT becomes $Adv_{\mathcal{A},Game_i}^{PBACS-PECIIoT} = Pr[Succ_{Game_i}^{\mathcal{A}}]$. The games are now defined as follows.

- **Game$_0^{\mathcal{A}}$**: Under this game, the adversary $\mathcal{A}$ plays a real attack under the ROR model for the initial game $Game_0^{\mathcal{A}}$. Prior to beginning of the game $Game_0^{\mathcal{A}}$, $\mathcal{A}$ needs to pick a random bit c. Therefore, the advantage of $Game_0^{\mathcal{A}}$ is then

$$Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) = |2Adv_{\mathcal{A},Game_0}^{PBACS-PECIIoT} - 1|. \tag{5.1}$$

- **Game$_1^{\mathcal{A}}$**: In this game, $\mathcal{A}$ applies the eavesdropping attack to derive the session key for a particular session. $\mathcal{A}$ performs the *Execute* query to intercept all the communicated messages $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS_{SD_i}\}$, $Msg_{AC2} = \{RID_{GN_j}, Sig_{GN_j}, RS_{GN_j}, y_{GN_j}, TS_{GN_j}\}$ and $Msg_{AC3} = \{SKV_{SD_i,GN_j}, TS'_{SD_i}\}$ during the access control phase (ACP) between SD_i and GN_j mentioned in Section 5.4.2. After that, $\mathcal{A}$ may try to generate the session key $SK_{SD_i,GN_j} = h(DHK_{SD_i,GN_j} || z_{SD_i})$, where $DHK_{SD_i,GN_j} = h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS_{SD_i}) \cdot RS_{GN_j}$ and $z_{SD_i} = y_{GN_j} \oplus h(DHK_{SD_i,GN_j} || TS_{SD_i} || TS_{GN_j})$. Without knowledge of the long term secrets $\{TC_{SD_i}, pr_{SD_i}\}$ and $\{TC_{GN_j}, pr_{GN_j}\}$, $\mathcal{A}$ cannot succeed to derive the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$. As the credentials are protected by the "cryptographic hash function $h(\cdot)$", $\mathcal{A}$ will be unable to derive the session key even by executing the *Reveal* and *Test* queries. Therefore, the games **Game$_1^{\mathcal{A}}$** and **Game$_0^{\mathcal{A}}$** are both indistinguishable under such an eavesdropping attack. The following outcome is then produced:

$$Adv_{\mathcal{A},Game_1}^{PBACS-PECIIoT} = Adv_{\mathcal{A},Game_0}^{PBACS-PECIIoT}. \tag{5.2}$$

- **Game$_2^{\mathcal{A}}$**: In this game, the adversary $\mathcal{A}$ plays an active attack. $\mathcal{A}$ simulates the *hash* and *CorruptSD* queries and tries to solve computational ECDDHP problem. $\mathcal{A}$ needs to obtain $DHK_{SD_i,GN_j} = h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS_{SD_i}) \cdot RS_{GN_j}$ $(= DHK_{GN_j,SD_i})$ to derive the session key SK_{SD_i,GN_j}. Assume that $\mathcal{A}$ hijacks all the transmitted message $\{Msg_{AC1}, Msg_{AC2}, Msg_{AC3}\}$. Thus, $\mathcal{A}$ knows the values RS_{SD_i} and RS_{GN_j}. From RS_{SD_i} and RS_{GN_j}, $\mathcal{A}$ may try to compute the secret values $h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS_{SD_i})$ and $h(TC_{GN_j} || rs_{GN_j} || pr_{GN_j} || RID_{GN_j} || TS_{GN_j})$, respectively. However, to derive these secrets credentials, $\mathcal{A}$ needs to know the long term secrets $\{TC_{SD_i},$

pr_{SD_i}, TC_{GN_j}, $pr_{GN_j}\}$, which becomes difficult problem due to solving ECDDHP. Moreover, the secrets are enclosed by a "one-way collision-resistant hash function $(h(\cdot))$". In addition, $\mathcal{A}$ will execute $CorruptSD$ to extract all the secret credentials $\{RID_{SD_i}$, TC_{SD_i}, $(pr_{SD_i}$, $Pub_{SD_i})$, $h(\cdot)$, $E_q(\alpha, \beta)$, $G_k\}$, but he/she has no knowledge about the random secrets (short term secrets)$\{rs_{SD_i}, rs_{GN_j}\}$. If $\mathcal{A}$ is aware of the long term secrets as well as short term, then only he/she gets the session key SK_{SD_i,GN_j} $(= SK_{GN_j,SD_i})$. Therefore, the games $\mathbf{Game_2^{\mathcal{A}}}$ and $\mathbf{Game_1^{\mathcal{A}}}$ are indistinguishable if we exclude the *hash* and $CorruptSD$ queries in $\mathbf{Game_2^{\mathcal{A}}}$. The birthday paradox result on "one-way collision-resistant hash function $(h(\cdot))$" and ECDDHP will result in the following relation:

$$|Adv_{\mathcal{A},Game_1}^{PBACS-PECIIoT} - Adv_{\mathcal{A},Game_2}^{PBACS-PECIIoT}| \leq \frac{q_h^2}{2|hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t). \tag{5.3}$$

Since all the games have been executed by $\mathcal{A}$, and it is "only remaining for $\mathcal{A}$ to correctly guess a bit to win the game $Game_2^{\mathcal{A}}$", we have,

$$Adv_{\mathcal{A},Game_2}^{PBACS-PECIIoT} = \frac{1}{2}. \tag{5.4}$$

Eq. (5.1) gives

$$\frac{1}{2}.Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) = |Adv_{\mathcal{A},Game_0}^{PBACS-PECIIoT} - \frac{1}{2}|. \tag{5.5}$$

Eq. (5.3) leads to the following inequality using Eq. (5.5):

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) &= |Adv_{\mathcal{A},Game_0}^{PBACS-PECIIoT} - Adv_{\mathcal{A},Game_2}^{PBACS-PECIIoT}| \\ &= |Adv_{\mathcal{A},Game_1}^{PBACS-PECIIoT} - Adv_{\mathcal{A},Game_2}^{PBACS-PECIIoT}| \\ &\leq \frac{q_h^2}{2|hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t). \end{aligned} \tag{5.6}$$

Hence, we have the final result: $Adv_{\mathcal{A},ACP}^{PBACS-PECIIoT}(t) \leq \frac{q_h^2}{|hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t)$. □

Remark 5.4. *If $Adv_{\mathcal{A},KMP}^{PBACS-PECIIoT}(t)$ be the advantage of an adversary $\mathcal{A}$ running in polynomial time t in order to derive the pairwise secret key SK_{GN_j,ES_l} $(= SK_{ES_l,GN_j})$ established between GN_j and ES_l in a particular session during the key management phase (KMP) for the proposed PBACS-PECIIoT, similar to Theorem 5.3, we also have:*

$$Adv_{\mathcal{A},KMP}^{PBACS-PECIIoT}(t) \leq \frac{q_h^2}{|hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t).$$

5.5.3 Informal security analysis

In the following propositions, through hueristic security analysis, we show that the proposed scheme (PBACS-PECIIoT) is secure against various other attacks.

Proposition 5.1. *The proposed PBACS-PECIIoT is secure against replay attack.*

Proof. In PBACS-PECIIoT, during the access control phase described in Section 5.4.2 between a smart device SD_i and its gateway node GN_j, the communicated messages Msg_{AC1}, Msg_{AC2}, and Msg_{AC3} have both random nonces and current timestamps. The freshness of the messages is provided by checking the timestamps. Similarly, for the key management among GN_j and its associated edge server ES_l described in Section 5.4.3 the communicated messages Msg_{KM1} and Msg_{KM2} are also having random numbers and current timestamps. Thus, the receivers can easily detect the old replayed messages that are re-transmitted by an adversary by validating the attached timestamps of the messages. Therefore, PBACS-PECIIoT is resilient against "replay attack". □

Proposition 5.2. *The proposed PBACS-PECIIoT is secure against Man-in-the-Middle (MiTM) attack.*

Proof. Suppose an adversary $\mathcal{A}$ eavesdrops the access control request message $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS_{SD_i}\}$ and tries to send another valid message, say Msg^*_{AC1} to the receiver GN_j. To achieve this goal, $\mathcal{A}$ can select a random number $rs^*_{SD_i} \in Z^*_q$ and timestamp $TS^*_{SD_i}$ on the fly, and then calculate $RS^*_{SD_i} = h(TC_{SD_i} || rs^*_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS^*_{SD_i}) \cdot G_k$. Without knowledge of the temporal credential TC_{SD_i} and permanent secret pr_{SD_i}, $\mathcal{A}$ can not compute valid $RS^*_{SD_i}$ and other valid signature $Sig^*_{SD_i}$ for Msg^*_{AC1}. Similarly, by intercepting the messages Msg_{AC2}, Msg_{AC3}, Msg_{KM1} and Msg_{KM2}, without temporal credentials and permanent secret, $\mathcal{A}$ can modify them on the fly. PBACS-PECIIoT is then resilient against "MiTM attack". □

Proposition 5.3. *The proposed PBACS-PECIIoT is resilient against impersonation attacks.*

Proof. Assume an adversary $\mathcal{A}$ plays as a legitimate smart device and tries to communicate with the gateway node by creating a valid message $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS^*_{SD_i}\}$. For successful attack, $\mathcal{A}$ can pick a random secret $rs_{SD_i} \in Z^*_q$ and timestamp $TS^*_{SD_i}$ to calculate $RS_{SD_i} = h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS^*_{SD_i}) \cdot G_k$. Since $\mathcal{A}$ has no idea about secrets TC_{SD_i} and pr_{SD_i}, $\mathcal{A}$ can not compute valid Msg_{AC1}. Similarly, it is also a "computationally impossible task" for $\mathcal{A}$ to construct other valid messages Msg_{AC2}, Msg_{AC3},

Msg_{KM1} and Msg_{KM2}. This means that PBACS-PECIIoT is secure against "smart device, gateway node and edge server impersonation attacks". □

Proposition 5.4. *The proposed PBACS-PECIIoT is resilient against privileged-insider attack.*

Proof. During the registration phase, RA_k does registration of all the entities (SD_i, GN_j, and ES_l) without providing any registration information from the entities. Instead, RA_k deletes all the secrets information (for example, temporal credentials and private keys) after the credentials are stored in the memory of the registering parties after successful registration prior to their deployment in a particular IIoT application. An adversary, being a privileged-insider user of any RA_K, can not then obtain any pre-loaded secret credentials of the deployed entities. Hence, PBACS-PECIIoT is resilient against "privileged-insider attack". □

Proposition 5.5. *The proposed PBACS-PECIIoT is resilient against physical IoT smart device capture attack.*

Proof. Due to existence of an unethical territory, there is a high chance that an adversary $\mathcal{A}$ can physically capture few IoT smart devices SD_i, and extract their stored credentials $\{RID_{SD_i}, TC_{SD_i}, (pr_{SD_i}, Pub_{SD_i}), h(\cdot), E_q(\alpha, \beta), G_k\}$ by applying the "power analysis attacks" [141]. However, the stored credentials are unique and different for all smart devices SD_i. Therefore, it is not possible for $\mathcal{A}$ to establish the session keys between a non-compromised SD_i and its respective GN_j. This circumstance is known as "unconditionally secure against smart device capture attack". As a result, PBACS-PECIIoT is secure against "physical vehicle capture attack". □

Proposition 5.6. *The proposed PBACS-PECIIoT protects Ephemeral secret Leakage (ESL) attack.*

Proof. During the access control process between SD_i and GN_j, they establish a common session key $SK_{SD_i,GN_j} = h(DHK_{SD_i,GN_j} || z_{SD_i})$ $(= SK_{GN_j,SD_i})$ where $DHK_{SD_i,GN_j} = h(TC_{SD_i} || rs_{SD_i} || pr_{SD_i} || RID_{SD_i} || TS_{SD_i}) \cdot RS_{GN_j}$. Similarly, during key management phase between GN_j and ES_l, a common session key is established as $SK_{GN_j,ES_l} = h(DHK_{GN_j,ES_l} || g_j(RID_{GN_j}, RID_{ES_l}))$ $(= SK_{ES_l,GN_j})$, where $DHK_{GN_j,ES_l} = h(RID_{GN_j} || rs_{GN_{j1}} || TC_{GN_j} || TS_{GN_{j1}} || pr_{GN_j}) \cdot RS_{ES_{l2}}$. In both the scenarios, in order to calculate DHK_{SD_i,GN_j} and DHK_{GN_j,ES_l} the short term (random nonces) and long term secrets (temporal credentials and private keys) are necessary. Since in every session the session keys are unique and distinct, even through a session key is compromised in a particular session it does not affect

on the session (secret) keys established in other sessions. PBACS-PECIIoT is then secure against "session-temporary information attack" and it also provides the "perfect forward and backward secrecy" goals at the same time. □

Proposition 5.7. *The proposed PBACS-PECIIoT provides block verification in Blockchain.*

Proof. In PBACS-PECIIoT, suppose a verifier $\mathcal{V}$ wants to verify a given block, say $Block_k$ in the blockchain. To successfully verify $Block_k$, $\mathcal{V}$ requires computation of "Merkle tree root ($MrkTR$)" on encrypted transactions and "current block hash ($CurBH$)" on all the entities in $Block_k$. If $MrkTR^* = MrkTR$ and $CurBH^* = CurBH$, $\mathcal{V}$ further validates Sig_{Block_k} using "ECDSA signature verification algorithm" with the public key Pub_{ES_l} of ES_l. Since $\mathcal{V}$ verifies all the $MrkTR$, $CurBH$ and Sig_{Block_k}, it is quite hard for an adversary to tamper the block $Block_k$ in the blockchain. If all the validations are successful, $\mathcal{V}$ accepts $Block_k$ as a valid block in the blockchain. □

Proposition 5.8. *The proposed PBACS-PECIIoT protects transaction privacy leakage.*

Proof. In blockchain, the user behavior can be traceable and it is important to preserve the transaction privacy of the users. A transaction in public blockchain may contain sensitive information and leakage of such critical data is a serious concern. Also, it is important to note that the input transaction should not be linked to its corresponding outputs. The "Bitcoin" and "Zcash" use one-time account to received cryptograms/puzzles. A secret key of user can be used within it so that an attacker cannot derive whether the same transaction contains a user's credential. Moreover, a common wallet may also leakage some vital information of the user. In the proposed PBACS-PECIIoT, due to the private blockchain criteria, the transactions in a block are encrypted with the help of public key of the corresponding edge server Pub_{ES_l}. Therefore, the privacy of the transactions are fulfilled in PBACS-PECIIoT. □

Proposition 5.9. *The proposed PBACS-PECIIoT protects selfish mining attack.*

Proof. Selfish mining attack is introduced by Eyal *et al.* in 2014 [75]. In a selfish mining attack [75], an attacker may misuse the computation power and steal the inappropriate rewards from the legitimate miners [19]. The attackers in the selfish mining may aim to retain the large private chain as compared to the public branch so that they can individually hold and dominate to add the additional new blocks. Thus, the selfish miners can obtain more blocks and have a competitive advantage over legitimate miners. This strategy has been extensively mentioned in Bitcoin, but very few attentions have been given to address it. Davidson *et al.*

[62] mentioned how the selfish mining can increment the earning of the miners for a larger collection of cryptocurrencies. In PBACS-PECIIoT, we have considered private blockchain and the mining is done by the P2P edge nodes which are treated as semi-trusted. Therefore, selfish mining attack would be hard to perform in the proposed system. □

Proposition 5.10. *The proposed PBACS-PECIIoT is resilient against balance attack.*

Proof. In this attack, an attacker tries to introduce a delay network communication between a valid range of subgroups consisting of similar mining power capabilities to execute the transactions. However, the miner needs to mine sufficient blocks to assure the subtree of another subgroup is equally essential as compared to the transaction subgroups. Moreover, an attacker can collect the transactions, which are not committed, in order to form a block and it has immense possibility of exceeding the subtree which consists of the transactions. In PBACS-PECIIoT, the individual edge server is connected with each application and it is semi-trusted in the private blockchain. As a result, it is difficult to create a separate chain and mine sufficient blocks into the blockchian. Hence, the balance attack is eliminated in the proposed PBACS-PECIIoT. □

Proposition 5.11. *The proposed PBACS-PECIIoT is resilient against Sybil attack.*

Proof. In this attack, an attacker can damage the reputation system by forging the identities (i.e. fake users' accounts) in the P2P network and use them to achieve the extremely huge domination in the network for making the legitimate entities in minority. Such virtual nodes or illegitimate nodes can then perform like genuine nodes to establish disproportionately huge influence on the P2P network. These may lead to various other attacks, such as "Denial-of-Service (DoS)" and "Distributed Denial-of-Service (DDoS)" attacks. However, it is required to verify or authenticate such nodes and the identities prior to joining the network. In PBACS-PECIIoT, if an edge server behaves like an attacker and tries to perform Sybil attack, it can not dominate the entire network and make the legitimate entities in minority. Therefore, the Sybil attack is resisted in the proposed PBACS-PECIIoT. □

5.6 Formal security verification using AVISPA: simulation study

The "AVISPA tool (Automated Validation of Internet Security Protocols and Applications)" provides a "modular and expressive formal language for specifying security protocols and prop-

```
%%% Registration Authority (RA )
role registrationauthority(RAk, SDi, GNj: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: RAk
played_by RAk
def=
local State: nat,
SKraksdi, SKrakgnj: symmetric_key,
F, Add, Poly: hash_func,
Gk,MKrak, IDsdi,IDgnj : text,
Prsdi,Pubsdi, RIDsdi, TCsdi, RTSsdi: text,
Prgnj, Pubgnj, RIDgnj, TCgnj, RTSgnj : text
const sp1, sp2: protocol_id
init  State := 0
transition

% smart device registration
1. State = 0  /\ Rcv(start) =|>
State' := 1   /\ Prsdi'  := new()
/\ Pubsdi' := F(Prsdi'.Gk)
/\ RIDsdi' := H(IDsdi.MKrak)
/\ TCsdi'  := H(RIDsdi'.MKrak.RTSsdi.Prsdi')
% Store registration message to IIoT memory
/\ Snd({RIDsdi'.TCsdi'.Prsdi'}_SKraksdi)
/\ secret({Prsdi',MKrak,RTSsdi,IDsdi}, sp1, {RAk,SDi})
%%Gatewaynode registration phase
/\ Prgnj'   := new()
/\ Pubgnj'  := F(Prgnj'.Gk)
/\ RIDgnj'  := H(IDgnj.MKrak)
/\ TCgnj'   := H(IDgnj.RTSgnj.MKrak)
% Store registration message to Gateway Node's memory
/\ Snd({RIDgnj'.TCgnj'.Prgnj'}_SKrakgnj)
/\ secret({Prgnj',MKrak,RTSgnj,IDgnj}, sp2, {RAk,GNj})
end role
```

Figure 5.6: HLPL role specification for the *RA* in Case 1

erties, known as the High-Level Protocol Specification Language (HLPSL)" and integrates various back-ends which help in implementing a "variety of automatic analysis techniques ranging from protocol falsification (by finding an attack on the input protocol) to abstraction-based verification methods for infinite numbers of sessions" [202]. AVISPA contains four backends, namely a) "On-the-Fly Model-Checker (OFMC)", b) "Constraint-Logic-based At-

```
%%% smart device SDi
role smartdevice(RAk, SDi, GNj: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Smart Device SDi
played_by SDi
def=
local State: nat,
SKraksdi: symmetric_key,
F, Add, Poly: hash_func,
RTSsdi, Prsdi, Rssdi, TSsdi, Rssdi1, Sigsdi, DHKsdignj,
Zsdi, SKsdignj, SKVsdignj, TSsdi1 : text,
Gk,MKrak, IDsdi,IDgnj, Prgnj, Rsgnj, TSgnj,
RTSgnj, Ygnj : text
const sp1, sp2, sdi_gnj_rssdi, sdi_gnj_tssdi, sdi_gnj_tssdi1,
gnj_sdi_rsgnj, gnj_sdi_tsgnj : protocol_id
init State := 0
transition
%%% receive registration message securely from the RA
1. State = 0 /\ Rcv({H(IDsdi.MKrak).H(H(IDsdi.MKrak).
MKrak.RTSsdi.Prsdi').Prsdi'}_SKraksdi) =|>
%{RIDsdi',TCsdi',prsdi'}
State' := 2 /\ secret({Prsdi',MKrak,RTSsdi}, sp1, {RAk,SDi})
%%%%% Access control phase
/\ Rssdi' := new() /\ TSsdi' := new()
/\ Rssdi1' := F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.
Prsdi'.H(IDsdi.MKrak).TSsdi').Gk)
/\ Sigsdi' := Add(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').H(H(IDsdi.MKrak).F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi')
%%% Send message Msg1 to GNj via public channel
/\ Snd(H(IDsdi.MKrak).Sigsdi'.Rssdi1'.TSsdi')
%% SDi has freshly generated rdr and TSdr for GN that are included in Msg1
/\ witness(SDi, GNj, sdi_gnj_rssdi, Rssdi')
/\ witness(SDi, GNj, sdi_gnj_tssdi, TSsdi')
%%% Receive message Msg2 from the GN via public channel
2. State = 2 /\ Rcv(H(IDgnj.MKrak)
.Add(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj').
H(H(IDgnj.MKrak).H(IDsdi.MKrak).F(Prgnj'.Gk).F(H(H(IDgnj.RTSgnj.
MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').Gk)).Ygnj').Prgnj)
.F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).
TSgnj').Gk) .xor(H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak)).
Add(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').H(H(IDsdi.MKrak).F(Prsdi'.Gk).
F(Prgnj'.Gk).TSsdi).Prsdi').TSgnj').H(F(H(H(IDgnj.RTSgnj.
MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.
Prsdi'.H(IDsdi.MKrak).TSsdi').Gk)).TSsdi'.TSgnj')).TSgnj') =|>
State' := 4 /\ TSsdi1' := new()
/\ DHKsdignj' :=F(F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').
Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').Gk)
.F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.
H(IDgnj.MKrak).TSgnj').Gk))
/\ Zsdi' := H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak))
.Add(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.
Prsdi'.H(IDsdi.MKrak).TSsdi').H(H(IDsdi.MKrak).
F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi').TSgnj')
/\ SKsdignj' := H(DHKsdignj'.Zsdi')
/\ SKVsdignj' := H(SKsdignj'.TSsdi1')
%%% Send message Msg3 to GNj via public channel
/\ Snd(SKVsdignj'.TSsdi1')
%% SDi has freshly generated TSsdi1 for GNj that are included in Msg3
/\ witness(SDi, GNj, sdi_gnj_tssdi1, TSsdi1')
% SDi acceptance of the values rsdi, tssdi for GNj by SDi
/\ request(GNj, SDi, gnj_sdi_rsgnj, Rsgnj')
/\ request(GNj, SDi, gnj_sdi_tsgnj, TSgnj')
end role
```

Figure 5.7: HLPL role specification for smart device SD_i in Case 1

```
role gatewaynode(RAk, SDi, GNj: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Smart Device GN
played_by GNj
def=
local State: nat,
SKrakgnj: symmetric_key,
F, Add, Poly: hash_func,
Gk, IDgnj, MKrak, RTSgnj, Prgnj, Rsgnj, TSgnj, RSgnj1,
DHKgnjsdi, Ygnj, Siggnj : text,
IDsdi, RTSsdi, Rssdi, Prsdi, TSsdi, TSsdi1  : text
const sp1, sp2, sdi_gnj_rssdi, sdi_gnj_tssdi, sdi_gnj_tssdi1,
gnj_sdi_rsgnj, gnj_sdi_tsgnj : protocol_id
init  State := 0
transition
%% GN registration phase
1. State = 0 /\ Rcv({H(IDgnj.MKrak).H(IDgnj.RTSgnj.MKrak).
Prgnj'}_SKrakgnj)  =|>
State' := 3 /\secret({Prgnj',MKrak,RTSgnj,IDgnj}, sp2, {RAk,GNj})
%% Access control Phase
%%% Received message Msg1 from the SDi via public channel
2. State = 3 /\ Rcv(H(IDsdi.MKrak).Add(H(H(H(IDsdi.MKrak).
MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').
H(H(IDsdi.MKrak).F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').Gk).TSsdi')=|>
State' := 5 /\ Rsgnj' := new() /\ TSgnj' := new()
/\ RSgnj1' := F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj').Gk)
/\ DHKgnjsdi' := F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').Gk))
/\ Ygnj' := xor(H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak))
.Add(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').H(H(IDsdi.MKrak).F(Prsdi'.Gk).
F(Prgnj'.Gk).TSsdi).Prsdi').TSgnj')
.H(DHKgnjsdi'.TSsdi'.TSgnj'))
/\ Siggnj' := Add(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.H(H(IDgnj.MKrak).H(IDsdi.MKrak).F(Prgnj'.Gk).DHKgnjsdi'.Ygnj').Prgnj)
%%% Send message Msg2 to the Smart Device via public channel
/\ Snd(H(IDgnj.MKrak).Siggnj'.RSgnj1'.Ygnj'.TSgnj')
%% Gateway Node(GNj) has freshly generated Rsgnj and TSgnj for smart device
/\ witness(GNj, SDi, gnj_sdi_rsgnj, Rsgnj')
/\ witness(GNj, SDi, gnj_sdi_tsgnj, TSgnj')
%%% Receive message Msg3 from the smart device via public channel
3. State = 5 /\ Rcv(H(H(F(F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').
Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').Gk)
.F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj').Gk))
.H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak)).Add(H(H(H(IDsdi.MKrak).
MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').
H(H(IDsdi.MKrak).F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi')
.TSgnj')).TSsdi1').TSsdi1') =|>
% GNj acceptance of the values (message Msg3) for GNj by SDi
State' := 7 /\ request(SDi, GNj, sdi_gnj_rssdi, Rssdi')
/\ request(SDi, GNj, sdi_gnj_tssdi, TSsdi')
/\ request(SDi, GNj, sdi_gnj_tssdi1, TSsdi1')
end role
```

Figure 5.8: HLPL role specification for gateway node GN_j in Case 1

```
role gatewaynode(RAk, SDi, GNj: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Smart Device GN
played_by GNj
def=
local State: nat,
SKrakgnj: symmetric_key,
F, Add, Poly: hash_func,
Gk, IDgnj, MKrak, RTSgnj, Prgnj, Rsgnj, TSgnj, RSgnj1,
DHKgnjsdi, Ygnj, Siggnj : text,
IDsdi, RTSsdi, Rssdi, Prsdi, TSsdi, TSsdi1  : text
const sp1, sp2, sdi_gnj_rssdi, sdi_gnj_tssdi, sdi_gnj_tssdi1,
gnj_sdi_rsgnj, gnj_sdi_tsgnj : protocol_id
init  State := 0
transition
%% GN registration phase
1. State = 0 /\ Rcv({H(IDgnj.MKrak).H(IDgnj.RTSgnj.MKrak).
Prgnj'}_SKrakgnj)  =|>
State' := 3 /\secret({Prgnj',MKrak,RTSgnj,IDgnj}, sp2, {RAk,GNj})
%% Access control Phase
%%% Received message Msg1 from the SDi via public channel
2. State = 3 /\ Rcv(H(IDsdi.MKrak).Add(H(H(H(IDsdi.MKrak).
MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').
H(H(IDsdi.MKrak).F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').Gk).TSsdi')=|>
State' := 5 /\ Rsgnj' := new() /\ TSgnj' := new()
/\ RSgnj1' := F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj').Gk)
/\ DHKgnjsdi' := F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').Gk))
/\ Ygnj' := xor(H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak))
.Add(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.
H(IDsdi.MKrak).TSsdi').H(H(IDsdi.MKrak).F(Prsdi'.Gk).
F(Prgnj'.Gk).TSsdi).Prsdi').TSgnj')
.H(DHKgnjsdi'.TSsdi'.TSgnj'))
/\ Siggnj' := Add(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj')
.H(H(IDgnj.MKrak).H(IDsdi.MKrak).F(Prgnj'.Gk).DHKgnjsdi'.Ygnj').Prgnj)
%%% Send message Msg2 to the Smart Device via public channel
/\ Snd(H(IDgnj.MKrak).Siggnj'.RSgnj1'.Ygnj'.TSgnj')
%% Gateway Node(GNj) has freshly generated Rsgnj and TSgnj for smart device
/\ witness(GNj, SDi, gnj_sdi_rsgnj, Rsgnj')
/\ witness(GNj, SDi, gnj_sdi_tsgnj, TSgnj')
%%% Receive message Msg3 from the smart device via public channel
3. State = 5 /\ Rcv(H(H(F(F(H(H(H(IDsdi.MKrak).MKrak.RTSsdi.Prsdi').
Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').Gk)
.F(H(H(IDgnj.RTSgnj.MKrak).Rsgnj'.Prgnj'.H(IDgnj.MKrak).TSgnj').Gk))
.H(Poly(H(IDgnj.MKrak).H(IDsdi.MKrak)).Add(H(H(H(IDsdi.MKrak).
MKrak.RTSsdi.Prsdi').Rssdi'.Prsdi'.H(IDsdi.MKrak).TSsdi').
H(H(IDsdi.MKrak).F(Prsdi'.Gk).F(Prgnj'.Gk).TSsdi).Prsdi')
.TSgnj')).TSsdi1').TSsdi1') =|>
% GNj acceptance of the values (message Msg3) for GNj by SDi
State' := 7 /\ request(SDi, GNj, sdi_gnj_rssdi, Rssdi')
/\ request(SDi, GNj, sdi_gnj_tssdi, TSsdi')
/\ request(SDi, GNj, sdi_gnj_tssdi1, TSsdi1')
end role
```

Figure 5.9: HLPL role specification for session, goal and environment in Case 1

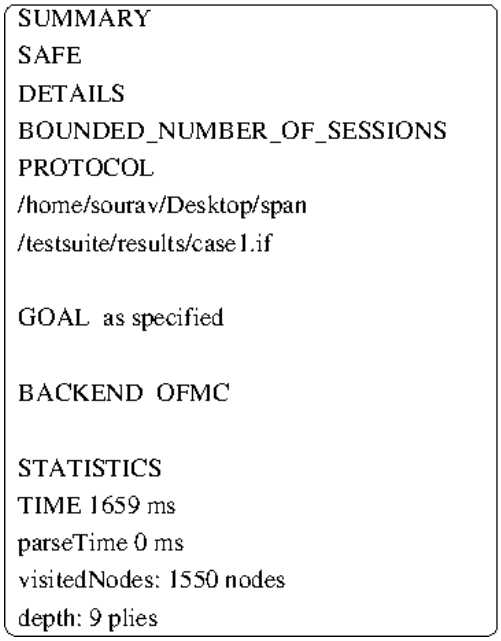

Figure 5.10: Simulation results of PBACS-PECIIoT for Case 1 under OFMC backend

Figure 5.11: Simulation results of PBACS-PECIIoT for Case 1 under CL-AtSe backend

tack Searcher (CL-AtSe)", c) "SAT-based Model Checker (SATMC)" and d) "Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)".

The proposed scheme (PBACS-PECIIoT) has been implemented under the HLPSL for two

scenarios:

- **Case 1:** It implements the registration and access control phases
- **Case 2:** It implements the registration and key management phases

In both the cases, we have the basic roles and the mandatory roles for the session and also for the goal and environment. In Case 1, the basic roles in HLPSL implementation for the RA, a smart device SD_i and a gateway node GN_j are shown in Figures 5.6, 5.7, and 5.8, respectively. The HLPL role specification for the session, goal and environment in Case 1 is also shown in Figure 5.9.

Since AVISPA implements the DY threat model [67] (as discussed in our threat model in Section 2.2), an intruder (i) always takes part of an active participating entity during the communication. Due to this, AVISPA has the ability to check whether a tested security protocol is resilient against "replay attack" and "man-in-the-middle attack". We have simulated Case 1 of PBACS-PECIIoT using the "SPAN, the Security Protocol ANimator for AVISPA" [18] under the widely-used OFMC and CL-AtSe backends. The simulation results for Case 1 demonstrated in Figures 5.10 and 5.11 clearly show that PBACS-PECIIoT is robust against both replay and man-in-the-middle attacks, under both OFMC and CL-AtSe backends.

In a similar way, in Case 2, the basic roles in HLPSL implementation for the RA, a gateway node GN_j, and an edge server ES_l are shown in Figures 5.12, 5.13, and 5.14, respectively. The HLPL role specification for the session, goal and environment in Case 1 is also shown in Figure 5.15. We have also simulated Case 2 of PBACS-PECIIoT using SPAN. The simulation results for Case 2 demonstrated in Figures 5.16 and 5.17 clearly show that PBACS-PECIIoT is robust against both replay and man-in-the-middle attacks, under both OFMC and CL-AtSe backends.

5.7 Experiments using MIRACL

We have done the testbed experiments for various cryptographic primitives with the help of widely-accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [5]. MIRACL is a C/C++ based Crypto SDK which is regarded by the software developers and cryptographers as the "gold standard open source SDK for elliptic curve cryptography (ECC)". In the following, we consider the following two scenarios:

The first platform that we have considered is for a server and the environment setting as "Ubuntu 18.04.4 LTS, with memory: 7.7 GiB, processor: Intel Core i7-8565U CPU @ 1.80GHz

```
role registrationauthority(RAk, GNj, ESl: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: RAk
played_by RAk
def=
local State: nat,
SKrakgnj, SKrakesl: symmetric_key,
F, Add, Poly: hash_func,
Gk,MKrak, Prgnj, Pubgnj, RIDgnj, IDgnj, TCgnj, RTSgnj : text,
Presl, Pubesl, RIDesl, IDesl : text
const sp1, sp2: protocol_id
init  State := 0
transition
%%Gatewaynode registration phase
1. State = 0  /\ Rcv(start) =|>
State' := 1   /\ Prgnj'   := new()
/\ Pubgnj'  := F(Prgnj'.Gk)
/\ RIDgnj'  := H(IDgnj.MKrak)
/\ TCgnj'   := H(IDgnj.RTSgnj.MKrak)
% Store registration message to Gateway Node's memory
/\ Snd({RIDgnj'.TCgnj'.Prgnj'}_SKrakgnj)
/\ secret({Prgnj',MKrak,RTSgnj,IDgnj}, sp1, {RAk,GNj})
%%Egde Server registration phase
/\ Presl'  := new()
/\ Pubesl' := F(Presl'.Gk)
/\ RIDesl' := H(IDesl.MKrak)
% Store registration message to Edge Server's memory
/\ Snd({RIDesl'.Presl'}_SKrakesl)
/\ secret({Presl',MKrak,IDesl}, sp2, {RAk,ESl})
end role
```

Figure 5.12: HLPL role specification for the RA in Case 2

$\times$ 8, OS type: 64-bit and disk: 966.1 GB". In the second platform, we have considered a smart device under the Raspberry PI 3 implementation where the environment setting is "Raspberry PI 3 B+ Rev 1.3, with CPU: 64-bit, Processor: 1.4 GHz Quad-core, 4 cores, Memory (RAM): 1GB, and OS: Ubuntu 20.04 LTS, 64-bit" [6]. In addition, all the experiments are run for 100 times for each cryptographic primitive under both the platforms, and we then considered the "maximum, minimum and average run-time (in milliseconds) for each cryptographic primitive".

We use the following cryptographic primitives for the testbed experiments. The notations T_h, T_{ecm}, T_{eca}, T_{me}, T_{bp}, T_m, T_a, and T_{senc}/T_{sdec} denote the time required to execute a "one-way cryptographic hash function", an "elliptic curve point (scalar) multiplication", an "elliptic

```
role gatewaynode(RAk, GNj, ESl: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Gateway Node GN
played_by GNj
def=
local State: nat,
SKrakgnj: symmetric_key,
F, Add, Poly: hash_func,
Gk, IDgnj, MKrak, RTSgnj, Prgnj, RSgnj1, TSgnj1, Rsgnj12, TCgnj, TSgnj, Siggnj : text,
Presl, IDesl, Rsesl2, TSesl2 : text
const sp1, sp2, gnj_esl_rsgnj1, gnj_esl_tsgnj1, esl_gnj_rsesl2, esl_gnj_tsesl2 : protocol_id
init State := 0
transition
%% GN registration phase
%%% Receive registration message securely from the RA
1. State = 0 /\ Rcv({H(IDgnj.MKrak).H(IDgnj.RTSgnj.MKrak).Prgnj'}_SKrakgnj) =|>
State' := 2 /\ secret({Prgnj',MKrak,RTSgnj,IDgnj}, sp1, {RAk,GNj})
%% Access control Phase
%%% Received message Msg1 from the GNj via public channel
/\ RSgnj1' := new() /\ TSgnj1' := new()
/\ Rsgnj12' := F(H(H(IDgnj.MKrak).RSgnj1'.H(IDgnj.RTSgnj.MKrak).TSgnj.Prgnj').Gk)
/\ Siggnj' := Add(F(H(H(IDgnj.MKrak).RSgnj1'.H(IDgnj.RTSgnj.MKrak).TSgnj.Prgnj').Gk).
H(H(IDgnj.MKrak).F(Prgnj'.Gk).F(Presl'.Gk).TSgnj1).Prgnj')
%%% Send message Msg1 to ESl via public channel
/\ Snd(H(IDgnj.MKrak).Rsgnj12'.Siggnj'.TSgnj1')
%% GN has freshly generated the values RSgnj1' and TSgnj1' for GN that are included in Msg1
/\ witness(GNj, ESl, gnj_esl_rsgnj1, RSgnj1')
/\ witness(GNj, ESl, gnj_esl_tsgnj1, TSgnj1')
%%% Receive message Msg2 from the ESl via public channel
2. State = 2 /\ Rcv(H(IDesl.MKrak).F(H(H(IDesl.MKrak).Presl'.Rsesl2'.TSesl2').Gk)
.Add(H(H(IDesl.MKrak).Presl'.Rsesl2'.TSesl2').H(F(Presl'.Gk).H(F(H(H(IDesl.MKrak).Presl'.
Rsesl2'.TSesl2').F(H(H(IDgnj.MKrak).RSgnj1'.TCgnj.TSgnj.Prgnj').Gk))
.Poly(H(IDesl.MKrak).H(IDgnj.MKrak))).TSesl2').Presl').TSesl2')=|>
State' := 4 /\ request(GNj, ESl, gnj_esl_rsgnj1, RSgnj1')
/\ request(GNj, ESl, gnj_esl_tsgnj1, TSgnj1')
end role
```

Figure 5.13: HLPL role specification for gateway node GN_j in Case 2

curve point addition", a "modular exponentiation operation", a "bilinear pairing operation", a "modular multiplication over $GF(q)$", a "modular addition over $GF(q)$", and a "symmetric encryption/decryption", respectively. We also considered a non-singular elliptic curve of the type: "$y^2 = x^3 + \alpha x + \beta \pmod q$" for "elliptic curve point addition and multiplication". Finally, the experimental results for various cryptographic primitives under a server platform and under the Raspberry PI 3 setting are provided in Table 5.2.

It is worth noticing that a Raspberry PI uses a "micro SD card" that has the capability to store both the system and data. If we compare a "micro SD card" to the "modern hard drives or solid-state drive (SSD) that are commonly found in computers (Desktops or Laptops)", the operations like reading and writing on the card are then quite slow in case of Raspberry PI [8]. This is why the results reported in Table 5.2 show the average time difference between

```
role edgeserver(RAk, GNj, ESl: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Edge Server
played_by ESl
def=
local State: nat,
SKrakesl: symmetric_key,
F, Add, Poly: hash_func,
Gk, MKrak, IDesl, Presl, Rsesl2, TSesl2, Rsesl23, DHKeslgnj, SKeslgnj, Sigesl2 : text,
IDgnj, RSgnj1, TSgnj, RTSgnj, Prgnj, TSgnj1 : text
const sp1, sp2, gnj_esl_rsgnj1, gnj_esl_tsgnj1, esl_gnj_rsesl2, esl_gnj_tsesl2 : protocol_id
init State := 0
transition
%% GN registration phase
1. State = 0 /\ Rcv({H(IDesl.MKrak).Presl'}_SKrakesl) =|>
State' := 3 /\ secret({Presl',MKrak,IDesl}, sp2, {RAk,ESl})
%% Access control Phase
%%% Received message Msg1 from the GNj via public channel
2. State=3/\Rcv(H(IDgnj.MKrak).F(H(H(IDgnj.MKrak).RSgnj1'.H(IDgnj.RTSgnj.MKrak).TSgnj.Prgnj').Gk)
.Add(F(H(H(IDgnj.MKrak).RSgnj1'.H(IDgnj.RTSgnj.MKrak).TSgnj.Prgnj').Gk).H(H(IDgnj.MKrak).
F(Prgnj').Gk).F(Presl'.Gk).TSgnj1).Prgnj').TSgnj1') =|>
State' := 5 /\ Rsesl2' := new() /\ TSesl2' := new()
/\ Rsesl23' := F(H(H(IDesl.MKrak).Presl'.Rsesl2'.TSesl2').Gk)
/\ DHKeslgnj' := F(H(H(IDesl.MKrak).Presl'.Rsesl2'.TSesl2').F(H(H(IDgnj.MKrak).
RSgnj1'.H(IDgnj.RTSgnj.MKrak).TSgnj.Prgnj').Gk))
/\ SKeslgnj' := H(DHKeslgnj'.Poly(H(IDesl.MKrak).H(IDgnj.MKrak)))
/\ Sigesl2' := Add(H(H(IDesl.MKrak).Presl'.Rsesl2'.TSesl2').H(F(Presl'.Gk).SKeslgnj'.
TSesl2').Presl')
%%% Send message Msg2 to the Gateway Node via public channel
/\ Snd(H(IDesl.MKrak).Rsesl23'.Sigesl2'.TSesl2')
%% Edge Server has freshly generated Rsesl2 and TSesl2 for Gateway node that are included in Msg2
/\ witness(ESl, GNj, esl_gnj_rsesl2, Rsesl2')
/\ witness(ESl, GNj, esl_gnj_tsesl2, TSesl2')
end role
```

Figure 5.14: HLPL role specification for edge server (ES_l) in Case 2

Table 5.2: Experimental results of cryptographic primitives on a server and a Raspberry PI 3 using MIRACL

Primitive	Average time (ms) under sever	Average time (ms) under Raspberry PI 3
T_{ecm}	0.674	2.288
T_{eca}	0.002	0.016
T_h	0.055	0.309
T_{me}	0.072	0.228
T_m	0.002	0.011
T_a	0.001	0.010
T_{bp}	4.716	32.084
T_{senc}	0.001	0.018
T_{sdec}	0.001	0.014

the server and Raspberry PI.

```
%%% Role for the session
role session (RAk, GNj, ESl: agent,
H: hash_func)
def=
local  Sn1, Sn2, Sn3, Rv1, Rv2, Rv3: channel (dy)
composition
registrationauthority (RAk,GNj, ESl, H, Sn1, Rv1)
/\ gatewaynode (RAk, GNj, ESl, H, Sn2, Rv2)
/\ edgeserver (RAk, GNj, ESl, H, Sn3, Rv3)
end role


%%% Role for the goal and environment
role environment()
def=
const rak, gnj, esl: agent,
h, f, add, poly: hash_func,
tsgnj1, tsesl2 : text,
sp1, sp2, gnj_esl_rsgnj1, gnj_esl_tsgnj1, esl_gnj_rsesl2,
esl_gnj_tsesl2 : protocol_id
intruder_knowledge = {rak, gnj, esl, h, f, add, poly, tsgnj1, tsesl2}
composition
session(rak, gnj, esl, h)
/\ session(rak, i, esl, h)
/\ session(rak, gnj, i, h)
end role
goal
%%% Confidentiality (privacy)
secrecy_of sp1, sp2
%%% Authentication
authentication_on gnj_esl_rsgnj1,gnj_esl_tsgnj1
authentication_on esl_gnj_rsesl2, esl_gnj_tsesl2
end goal
environment()
```

Figure 5.15: HLPL role specification for session, goal and environment in Case 2

5.8 Comparative study

In this section, we provide a detailed comparative study on "security and functionality features", "communication costs" and "computation costs" among the proposed PBACS-PECIIoT and other state-of-art schemes of Li *et al.* [121], Luo *et al.* [131] and Xue *et al.* [226].

Figure 5.16: Simulation results of PBACS-PECIIoT for Case 2 under OFMC backend

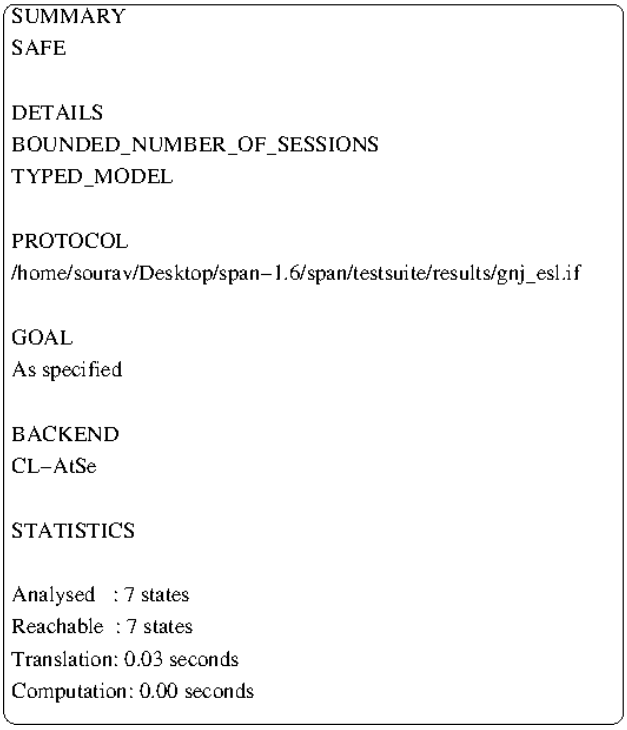

Figure 5.17: Simulation results of PBACS-PECIIoT for Case 2 under CL-AtSe backend

5.8.1 Communication costs comparison

In PBACS-PECIIoT, to evaluate the communication costs for the access control phase (Case 1) between SD_i and GN_j and for the key management phase (Case 2) among GN_j and ES_l, we consider only communication messages among them. It is assumed that a "random number", an "identity", a "one-way hash function (using SHA-256 hashing algorithm)", an "elliptic curve point $P \in E_q(\alpha, \beta)$" and a "timestamp" are 160, 160, 256, 320, and 32 bits, respectively.

In Case 1 of PBACS-PECIIoT, the communication costs for the messages $Msg_{AC1} = \{RID_{SD_i}, Sig_{SD_i}, RS_{SD_i}, TS_{SD_i}\}$, $Msg_{AC2} = \{RID_{GN_j}, Sig_{GN_j}, RS_{GN_j}, y_{GN_j}, TS_{GN_j}\}$ and $Msg_{AC3} = \{SKV_{SD_i,GN_j}, TS'_{SD_i}\}$ require (256 + 160 + 320 + 32) = 768 bits, (256 + 160 + 320 + 256 + 32) = 1024 bits, and (256 + 32) = 288 bits, which altogether demand 2080 bits. In Case 2 of PBACS-PECIIoT, the messages $Msg_{KM1} = \{RID_{GN_j}, RS_{GN_{j1}}, Sig_{GN_{j1}}, TS_{GN_{j1}}\}$ and $Msg_{KM2} = \{RID_{ES_l}, RS_{ES_{l2}}, Sig_{ES_{l2}}, TS_{ES_{l2}}\}$ needs equally (256 + 320 + 160 + 32) = 768 bits, which altogether require 1536 bits. The comparative study shown in Table 5.3 demonstrates that the communication costs for both Case 1 and Case 2 require less costs as compared to other schemes.

Table 5.3: Comparison of communication costs

Protocol	No. of messages	Total cost (in bits)
PBACS-PECIIoT (Case 1)	3	2080
PBACS-PECIIoT (Case 2)	2	1536
Li *et al.* [121]	4	5408
Luo *et al.* [131]	2	3040
Xue *et al.* [226]	5	9344

5.8.2 Computation costs comparison

Assume T_{poly} denotes the time required for "evaluation of an t-degree uni-variate polynomial". Based on the Horner's rule [110], evaluating an "t-degree uni-variate polynomial" requires "t modular multiplications" and "t modular additions", that is, $T_{poly} = tT_m + tT_a$. We have used the average time listed in Table 5.2 needed for various cryptographic primitives for a server. On the other side, we have used the average time listed in Table 5.2 needed for various cryptographic primitives for a smart device or user's mobile device under Raspberry PI 3.

In Case 1 of PBACS-PECIIoT, an IoT smart device SD_i requires the computation cost of $6T_h + 4T_{ecm} + T_{eca} \approx 11.022$ ms and a gateway note GN_j needs the computation cost of $7T_h + 4T_{ecm} + T_{eca} + T_{poly} \approx 6.083$ ms. In Case 2 of PBACS-PECIIoT, both GN_j and ES_l equally need the computation cost of $4T_h + 4T_{ecm} + T_{eca} + T_{poly} \approx 5.918$ ms. Here, we have considered $t = 1000$ to support "unconditional security" as suggested by Blundo *et al.* [33]. The comparative analysis on computation costs in PBACS-PECIIoT for both Case 1 and Case 2 shows that PBACS-PECIIoT needs comparable costs with other existing schemes that are tabulated in Table 5.4.

Table 5.4: Comparative computational costs analysis

Scheme	Smart device/User cost	Server cost (GN/ES)
PBACS-PECIIoT (Case 1)	$6T_h + 4T_{ecm} + T_{eca}$ ≈ 11.022 ms	$7T_h + 4T_{ecm} + T_{eca}$ $+tT_m \approx 6.083$ ms
PBACS-PECIIoT (Case 2)	–	$8T_h + 8T_{ecm} + 2T_{eca}$ $+2tT_m \approx 11.836$ ms
Li *et al.* [121]	$5T_{me} + 3T_h$ ≈ 2.067 ms	$4T_{me} + 3T_h + T_{bp}+$ $T_{ecm} + T_{eca} \approx 5.845$ ms
Luo *et al.* [131]	$T_{bp} + T_h$ ≈ 32.393 ms	$3T_{ecm} + 3T_{bp} + 3T_h+$ $T_{eca} + T_{me} \approx 16.409$ ms
Xue *et al.* [226]	$4T_{ecm} + T_{eca} + 5T_{bp}$ $+4T_h + 6T_{senc}/T_{sdec}$ ≈ 170.92 ms	$3T_h + 3T_{me} + 2T_{ecm}$ $+T_{eca} + T_{bp}$ ≈ 6.447 ms

5.8.3 Security and functionality features comparison

Various "security and functionality features" (FSF_1–FSF_{16}) are considered in comparative study among PBACS-PECIIoT and other schemes (see Table 5.5). It is evident that PBACS-PECIIoT provides better security features and more functionality attributes as compared to those for other schemes of Li *et al.* [121], Luo *et al.* [131] and Xue *et al.* [226]. Considering the comparative analysis on "communication and computation costs" and "security and functionality features" (FSF_1–FSF_{16}), we can say that PBACS-PECIIoT is much practical to be deployed for PEC in IIoT environment.

Since the fog servers are semi-trusted, the distributed databases with only timestamps can not help to fulfill all the security requirements such as insider attack, device physical cap-

ture attack, and most importantly session key security (ESL attack) under the CK-adversary model. However, the proposed PBACS-PECIIoT provides the security features as compared to the existing schemes.

Table 5.5: Comparison of functionality & security features

Feature	Li *et al.* [121]	Luo *et al.* [131]	Xue *et al.* [226]	PBACS-PECIIoT
FSF_1	✓	✓	✓	✓
FSF_2	✓	✓	✓	✓
FSF_3	✓	✓	✓	✓
FSF_4	×	×	✓	✓
FSF_5	✓	✓	×	✓
FSF_6	✓	✓	✓	✓
FSF_7	✓	×	✓	✓
FSF_8	N/A	✓	N/A	✓
FSF_9	×	×	×	✓
FSF_{10}	×	✓	×	✓
FSF_{11}	×	×	×	✓
FSF_{12}	×	×	×	✓
FSF_{13}	×	×	×	✓
FSF_{14}	×	×	×	✓
FSF_{15}	×	×	×	✓
FSF_{16}	×	×	×	✓

Note: FSF_1: "resistant to privileged insider attack"; FSF_2: "replay attack"; FSF_3: "man-in-the-middle attack"; FSF_4: "mutual authentication"; FSF_5: "key agreement"; FSF_6: "device/gateway node impersonation attack"; FSF_7: "resilience against device physical capture attack"; FSF_8: "edge server impersonation attack"; FSF_9: "session key security under the CK-adversary model"; FSF_{10}: "formal security verification using AVISPA tool"; FSF_{11}: "dynamic node addition phase"; FSF_{12}: "support to blockchain-based solution"; FSF_{13}: "transaction privacy leakage"; FSF_{13}: "selfish mining attack "; FSF_{13}: "balance attack "; FSF_{13}: " Sybil attack".

✓: "a scheme is secure or it assists a feature"; ×: "a scheme is insecure or it does not assist a feature"; N/A: "not applicable in a scheme".

5.9 Blockchain implementation

In this section, we present the practical implementation of our proposed PBACS-PECIIoT, and measure its performance in terms of computational time. The computational time is considered to measure the costs for a block addition and mined in the P2P network. The performance evaluation is considered using a reasonable amount of data for simulation. However, the proposed model can also handle a huge volume of data for an IIoT environment. Many discussions are for scalability issues in blockchain, but in real world scenario the lightning network [160] can be used to handle the high transactions volume. The lightning network is a layer 2 protocol which is specifically used to improve the scalability in blockchain network. The environment was considered for simulation with the following setting: "CPU Architecture: 64-bit, Processor: 2.60 GHz Intel Core i5-3230M, Memory: 8 GB, OS: Ubuntu 18.04.4 LTS".

In each block in the blockchain, we have the block version ($BlkV$), previous block hash ($PreBH$), Merkle tree root ($MrkTR$), industry type (IT_m), timestamp (TS_{ES_l}), creator of block ($CreB_{ID}$), public key of signer (Pub_{ES_l}), current block hash ($CurBH$), signature (Sig_{Block_k}), whose sizes are taken as 32, 256, 256, 32, 32, 160, 320, 256 and 320 bits, respectively. In addition, each encrypted transaction $E_{Pub_{ES_l}}(Tx_{t_i})$, $(i = 1, 2, \cdots, t_n)$, consists of two elliptic curve points and hence, it needs (320+320) = 640 bits. The total block size then turns out to be $1664 + 640t_n$ bits.

In order to measure the block generation time in the proposed PBACS-PECIIoT with respect to the block structure mentioned in Figure 5.5, we have considered the average computational time (in milliseconds) for hash function and ECDSA signature generation under the MIRACL library for a server setting platform (see Table 5.2). This is because each edge server is resource rich node in the network. Note that the time needed for an ECDSA signature generation is approximately $T_{ecm} + T_h$. In addition, we have also implemented the Merkle tree using SHA-256 hashing algorithm. Based on these results, an edge server can compute the block generation time. In Figure 5.18, a block generation time (in milliseconds) by an edge server is shown for various number of encrypted transactions containing in the block. The results show that the computational time increases when the number of encrypted transactions in a block also increases.

Now, the blockchain implementation has been performed using the node.js language with VSCODE 2019 with the voting-based consensus algorithm explained in Figure 3. The following three cases are taken:

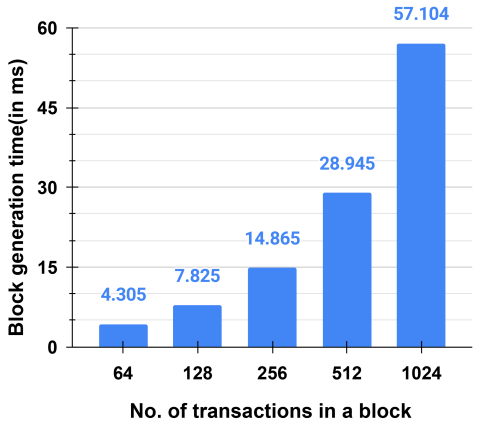

Figure 5.18: Block generation time (in milliseconds) by an edge server, ES_l

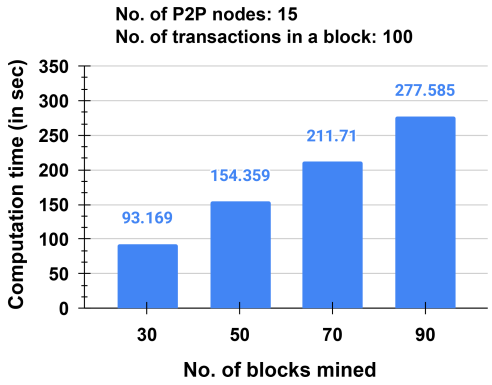

Figure 5.19: Blockchain simulation results in Case-I

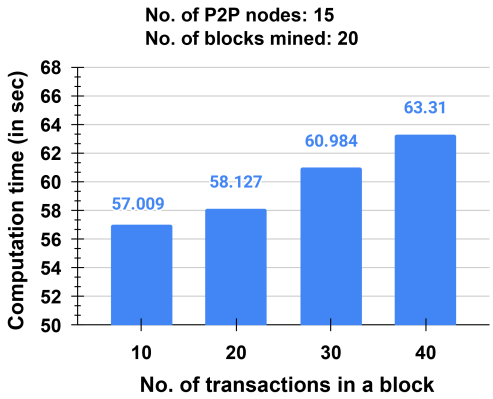

Figure 5.20: Blockchain simulation results in Case-II

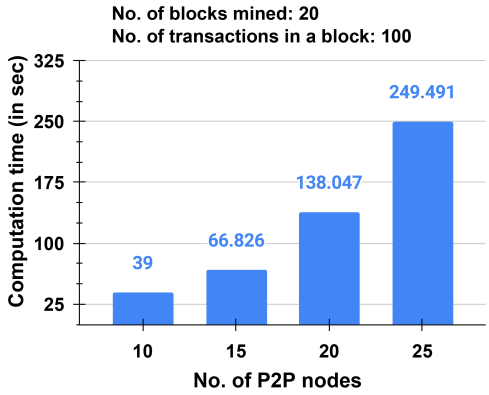

Figure 5.21: Blockchain simulation results in Case-III

- **Case-I**: We have considered the number of blocks mined versus the total computational time (in milliseconds) with the number of P2P nodes as 15 and the number of transactions per each block as 100. The blockchain simulation outcomes under this scenario are presented in Figure 5.19. It is observed that "as the number of blocks mined increases, the total computational time increases".

- **Case-II**: In this case, we have considered the number of transactions in per block versus the total computational time (in milliseconds). The number of blocked mined is fixed at 20, whereas the number of P2P nodes remains as in Case 1 as 15. Figure 5.20 presents the simulation results. It is worth noticing that the "total computational time increases as the number of transactions per block also increases".

- **Case-III**: In this case, we have considered the number of P2P nodes versus the total computational time (in milliseconds). Moreover, the number of blocked mined is fixed at 20 and the number of transactions in per block is also fixed as 100. We can observe from Figure 5.21 that the "total computational time increases with the increasing number of P2P nodes too".

5.10 Summary

We proposed a robust and efficient blockchain-based access control enabled blockchain solution for PEC in IIoT deployment (PBACS-PECIIoT). We considered private blockchain scenario due to strictly confidential and private data belonging to each IIoT application. The proposed PBACS-PECIIoT is not only secure against various potential attacks, but it also offers various

functionality features. The simulation results using the formal security verification under AVISPA automated software tool demonstrate that PBACS-PECIIoT is secure against passive and active attacks. Finally, a detailed comparative study reveals that PBACS-PECIIoT offers better "security features" and more "functionality features", requires low "communication costs" and comparable "computational costs" as compared to existing relevant recent schemes.

Chapter 6

Consortium Blockchain-Enabled Access Control in Edge Computing-Based Generic IoT Environment

In a consortium blockchain, lesser nodes are involved in comparison with the "public and private blockchain networks". Thus, it helps in processing the transactions at a much faster speed. Since the consortium blockchain is considered as a permissioned network, each entity requires a prior approval before its joining to an organization and it also works collectively across various organizations. In this chapter, we consider an edge computing-based generic Internet of Things (IoT) environment, whose structure well matches with the model of a cross-domain IoT deployment. Due to these issues, we propose a novel consortium blockchain-enabled access control scheme in edge computing-based generic IoT deployment as compared to the access control mechanisms that were proposed in Chapters 4 and 5.

6.1 Motivation

If an adversary can manipulate the genuine information, the transactions contained in a block in the blockchain will not be also genuine. The access control mechanism plays a very crucial role here, because the IoT devices require to send the information to their nearby gateway node and also the gateway node needs to send the data to its associated edge server(s) securely. As a result, we need to have a secure access control mechanism in edge computing based generic

IoT environment to make secure communication among various entities in the network. The edge servers in a peer-to-peer (P2P) edge servers network are considered as trusted and they are responsible for creating, verifying and adding the blocks in their local ledgers first using consensus algorithm. Later, the local ledgers maintained by the edge servers are then periodically updated in the blockchain's global ledger in order to avoid much burden at the blockchain center. Due to blockchain technology, once the blocks are added into the blockchain, these can not be further modified, updated or even deleted from the blockchain because each block contains previous block hash, Merkle tree root, signature on the transactions and also current block hash. Most of the access control protocols proposed in the literature in the IoT and also in resource-constrained wireless sensor networks environments are vulnerable to attacks, and they do not support blockchain feature in order to provide stronger security and more functionality features, such as block verification in blockchain, transparency, decentralization and immutability properties. This chapter attempts to design a novel access control protocol in edge computing based generic IoT environment where depending on the importance of the data in IoT applications, the data are encrypted in the block (private blockchain) or these are stored in unencrypted form in the block in the blockchain (public blockchain). There may be some applications where we need to have both private and public data to be stored in a block in the blockchain (consortium blockchain). Hence, we consider consortium blockchain access control mechanism to address these issues.

6.2 Research contributions

The main contributions made in the chapter as listed below:

- We design a new consortium blockchain-enabled access control scheme in edge computing based generic IoT environment (called CBACS-EIoT). CBACS-EIoT offers access control among IoT smart devices and their associated gateway nodes and also among the gateway nodes and edge servers. In addition, key management process among the edge servers and the cloud servers in the blockchain center.

- The blocks created by the edge nodes are mined and put in their respective local ledgers. The local ledger having blocks are then added in the global ledger maintained by the cloud servers in the blockchain center. All the secure communications among the IoT smart devices, gateway nodes, edge servers and cloud servers happen using their respective secret (session) keys.

- To assure that the proposed CBACS-EIoT is highly secure against various potential attacks needed for an IoT environment, the formal security and informal security analysis, and also the formal security verification using the broadly accepted AVISPA tool have been performed.

- A meticulous comparative analysis on security and functionality features, communication and computation overheads among the proposed CBACS-EIoT and existing schemes has been also performed to demonstrate the superiority of security and efficiency of CBACS-EIoT over other existing schemes.

6.3 System models

In this section, we discuss the network and threat models that are applied in our proposed scheme.

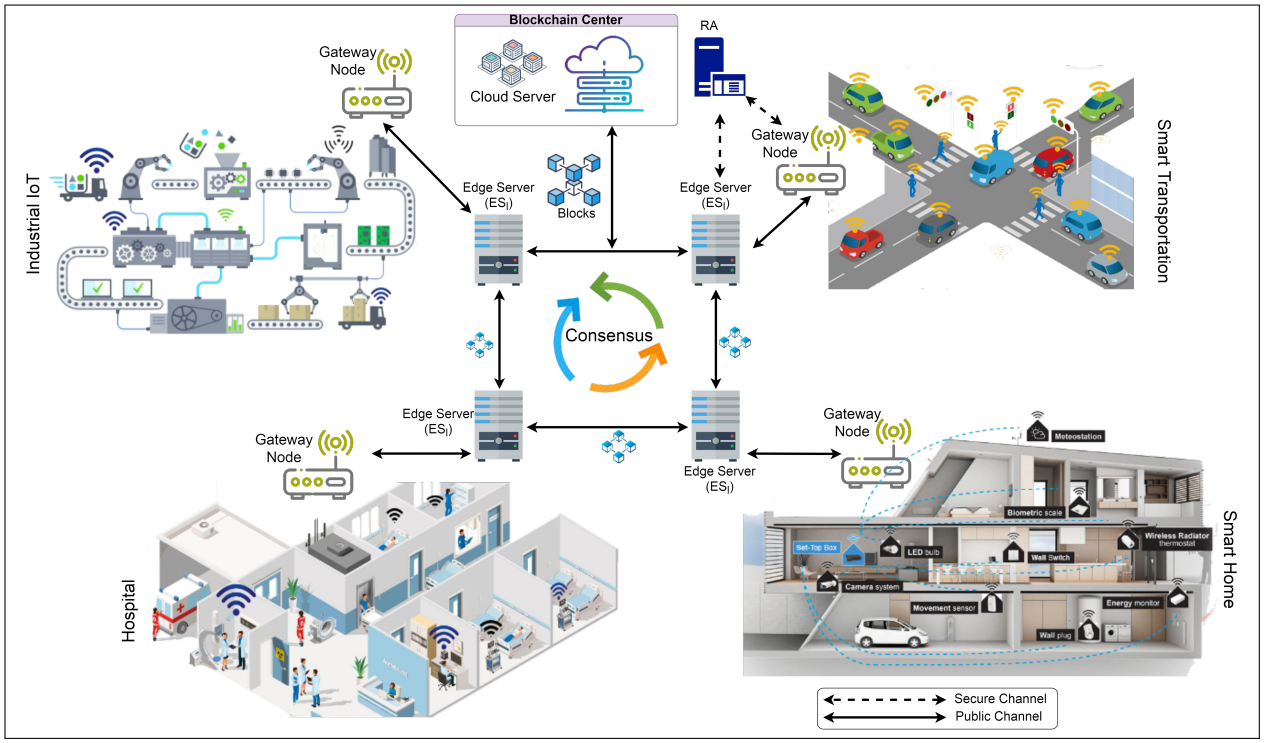

Figure 6.1: Blockchain-based network model for generic IoT network

6.3.1 Network model

The blockchain-based generic IoT network model is depicted in Figure 6.1. In this model, there exist different IoT applications, such as smart home, healthcare, industrial monitoring, smart vehicles and smart traffic management appliances, etc. For each application, there are several IoT smart devices (SD_i) are that installed in a proximity of their associated gateway node (GW_j). One or mode GW_j are associated with their nearby edge server ES_l.

A group of authorized edge servers form a peer-to-peer (P2P) network, called P2P ES network. The peers (nodes) in the P2P ES network are responsible for mining the blocks for adding them into their distributed *local ledgers* by applying some consensus algorithms. In this chapter, we apply the "Ripple Protocol Consensus Algorithm (RPCA)" [209]. After certain period of time, the local ledger is then updated in the corresponding existing global ledger that is maintained by the BC.

For forming the blocks by an edge server (ES_l), the data is first securely collected by the gateway node(s) and the transactions formed by the securely collected data are passed secretly to ES_l. The ES_l is then responsible for forming blocks and adding them in the local ledger. This detailed process is explained in our proposed scheme (see Section 6.4). Various roles of the involved entities in the network model are given below:

- *IoT smart device:* Several IoT smart devices can be installed or deployed in a particular application. The smart devices are typically responsible for gathering their surrounding information and transmit them to their nearby gateway nodes for further processing.

- *Gateway node:* A gateway node (GW) acts as an access point for a particular application. The data collected by the GW form various transactions, which are then forwarded to its associated edge sever (ES).

- *Edge server:* An edge server is a device that manages the flow of data at the boundary among two networks. It typically acts as a network entry (or exit) point, and is also responsible for various activities, such as "transmission", "processing", "routing", "filtering", "monitoring", "translation", "storage of information passing between networks", etc. In this chapter, we consider an edge server which is mainly responsible for collecting the transactions securely from its associated gateway nodes to form blocks, and then the blocks are mined to add them into its local ledger.

- *Blockchain center:* A blockchain center (BC) is a group of cloud-based servers, which is responsible for storing various data passed by the edge servers. In this chapter, we

consider the BC as an entity where all the blocks are added from the local ledgers of the edge servers to the global ledger of the BC in the blockchain.

- *Registration authority:* A registration authority (RA) is basically a fully trusted entity in the network, which is responsible for registering all the deployed IoT smart devices, the gateway nodes, edge servers and cloud servers in the BC. After successful registration, each entity is pre-loaded with proper credentials prior to its deployment or placement in the edge-based IoT environment.

6.3.2 Threat model

We apply the following threat model for this work in this chapter.

- Typically the widely-accepted "Dolev-Yao threat model (known as DY model)" [67] is applied in a threat model in general networking environment. We also apply the DY model in the IoT environment, because an adversary $\mathcal{A}$ can not only intercept the communication between two entities (i.e., smart devices, gateway nodes, edge servers and cloud servers), but can also insert malicious message contents in the transmitted messages apart from modifying and deleting the contents in the communicated messages.

- Recently, another model, known as the "Canetti and Krawczyk's model (CK-adversary model)" [37] is more powerful as compared to the DY model, because the adversary $\mathcal{A}$ can compromise the secret credentials, session keys and session states in a particular running session between two communicating parties in the network. Thus, it is extremely important that a designed protocol should have minimal effect on compromising past and future session keys established between two entities even if a currently running session is hijacked by $\mathcal{A}$.

- It is assumed that the end point entities like the IoT smart devices are trusted in the IoT environment. Furthermore, the blockchain center (BC) is treated as fully-trusted node, whereas the gateway nodes and edge servers are considered as semi-trusted entities in the IoT environment.

- Since the IoT smart devices can not be monitored 24×7 like environment, there is chance of physical capturing of some IoT smart devices in the network. Using the power analysis attacks as stated in [141], the extracted data from the memory of the captured IoT smart devices can be further utilized in mounting other attacks like impersonation attacks on other non-compromised IoT smart devices.

6.4 The proposed scheme: CBACS-EIoT

This section introduces a new proposed consortium blockchain-enabled access control scheme in edge computing based generic IoT environment, which we call it as CBACS-EIoT. CBACS-EIoT is based on a generic IoT based architecture with blockchain-enabled technology, where based on a particular type of application it will be decided whether the block content needs to be protected (encrypted) or not. For instance, in healthcare, smart home and surveillance applications, the transactions containing in a block needs to be encrypted so that other unauthorized entities can not see the information stored in that block. On the other hand, in smart transportation system and smart city environments the transactions (information) in a block need not to be protected while the modification/updation of those information must be infeasible task by unauthorized entities. There may be some applications in the IoT environment, where we may require to protect the transactions in the blocks (private blocks) and also to store the transactions without encryption (public blocks). Due to this, in this chapter we consider the consortium blockchain where both public and private blocks can be stored together in a blockchain that will be finally maintained by the blockchain center (BC) which is shown in Figure 6.1. The list of symbols with their importance supplied in Table 6.1 for describing and analyzing the proposed CBACS-EIoT.

For strong replay attack protection in the proposed CBACS-EIoT, we plan to utilize both the random nonces and current system timestamps. The timestamp is typical used in many authentication and access control protocols in several networks, including IoT, wireless sensor networks, smart grids, smart homes, Internet of Drones (IoD), Internet of Vehicles (IoV), etc. [39, 55, 59, 113, 137, 188, 211, 217, 218].

An access control mechanism in IoT environment deals with the two important functions [60]: a) *node (device) authentication* and b) *key establishment*. Access control mechanisms are broadly of two types: a) "certificate-less" and b) "certificate-based". This chapter develops a certificate-less access control mechanism to assist real-time data access by the gateway nodes from their respective IoT smart devices based on their application types (i.e., battlefield, healthcare, smart home, smart grid, and smart agriculture) in order make several transactions of building blocks (private, public or consortium) and then adding the created blocks in the blockchain.

Table 6.1: List of symbols with their importance

Symbol	Significance
$E_q(u, v)$	A non-singular elliptic curve of the form: "$y^2 = x^3 + ux + v \pmod q$ with $4u^3 + 27v^2 \neq 0 \pmod q$", two constants $u, v \in Z_q = \{0, 1, 2, \ldots, q-1\}$, and q is a large prime so that elliptic curve discrete logarithm problem (ECDLP) is intractable
G	A base point in $E_q(u, v)$ whose order is n_G as big as q
$k \cdot G$	Elliptic curve point (scalar) multiplication; $k \cdot G = G + G + \cdots G\,(k \text{ times})$, $k \in Z_q^*$
$Q + R$	Elliptic curve point addition; $Q, R \in E_q(u, v)$
$u * v$	Ordinary multiplication of two numbers $u, v \in Z_q$
RA, mk_{RA}	Trusted registration authority and its master key
SD_i, ID_{SD_i}	i^{th} IoT smart device and its associated identity
TID_{SD_i}, PID_{SD_i}	Unique temporary and pseudo identities of SD_i, respectively
mk_{SD_i}	Unique master key of SD_i
RTS_{SD_i}	Registration timestamp of SD_i
(k_{SD_i}, Pub_{SD_i})	Private-public key pair of SD_i, where $Pub_{SD_i} = k_{SD_i} \cdot G$
GW_j	j^{th} gateway node associated with an application (i.e., healthcare, smart-home, surveillance, smart transportation, smart parking management system)
ID_{GW_j}, mk_{GW_j}	Unique identity & master key of GW_j
(k_{GW_j}, Pub_{GW_j})	Private-Public key pair of GW_j; $Pub_{GW_j} = k_{GW_j} \cdot G$
TID_{GW_j}, PID_{GW_j}	Unique temporary and pseudo identities of GW_j, respectively
ES_l	l^{th} Edge server connected with one or more gateway node(s) GW_j
ID_{ES_l}, mk_{ES_l}	Identity and master key of ES_l, respectively
(k_{ES_l}, Pub_{ES_l})	Private-public key pair of ES_l, where $Pub_{ES_l} = k_{ES_l} \cdot G$
TID_{ES_l}, PID_{ES_l}	Unique temporary and pseudo identities of ES_l, respectively
CS_k	k^{th} cloud server in blockchain center connected with one or more ES_l
ID_{CS_k}, mk_{CS_k}	Identity and master key of CS_k, respectively
(k_{CS_k}, Pub_{CS_k})	Private-public key pair of CS_k, where $Pub_{CS_k} = k_{CS_k} \cdot G$
TID_{CS_k}, PID_{CS_k}	Unique temporary and pseudo identities of CS_k, respectively
n_{sd_j}	Number of IoT smart devices belonging to GW_j
n_{es}	Number of edge servers in the IoT environment
n_{cs}	Number of cloud servers in the blockchain center
n_{gw_l}	Number of gateway nodes belonging to an ES_l
$f(x, y)$	Symmetric bivariate t_1-degree polynomial over the Galois field $GF(q)$: $f(x, y) = \sum_{i=0}^{t_1} \sum_{j=0}^{t_1} a_{i,j} x^i y^j$, where $a_{i,j} \in Z_q$, is selected for each edge server ES_l and its associated cloud server(s) CS_k
$g_j(x, y)$	j^{th} t_2-degree symmetric bivariate polynomial $g_j(x, y) = \sum_{m=0}^{t_2} \sum_{n=0}^{t_2} b_{m,n} x^m y^n$ over $GF(q)$, where $b_{m,n} \in Z_q$ is randomly selected for each GW_j and its associated edge server(s) ES_l

6.4.1 Setup phase

This phase is executed by the trusted registration authority (RA) to selected all the private and public related credentials. The following steps are performed by the RA:

- Step S_1: The RA first picks a "non-singular elliptic curve" $E_q(u, v)$ of the form: "$y^2 = x^3 + ux + v \pmod q$ with $4u^3 + 27v^2 \neq 0 \pmod q$", where q is a large prime so that "elliptic curve discrete logarithm problem (ECDLP)" becomes intractable, with a "point at infinity" or "zero point" $\mathcal{O}$. Next, the RA selects a base point G in $E_q(u, v)$ whose order is n_G as big as q, that is, $n_G \cdot G = G + G + \cdots + G$ (n_G times) $= \mathcal{O}$, where $k \cdot G$ is known as the "elliptic curve point (scalar) multiplication" of the point $G \in E_q(u, v)$ and $k \in Z_q^* = \{1, 2, \cdots, q-1\}$.

- Step S_2: The RA selects a "cryptographic one-way hash function $h(\cdot)$". For example, we use the Secure Hash Standard (SHA-256) hashing algorithm [140] to provide sufficient security in the blockchain technology, which takes an arbitrary length string as input and outputs 256-bit hash value as message digest. In addition, the RA also picks its own master key mk_{RA}.

- Step S_3: Finally, the RA makes the information $\{E_q(u, v), G, h(\cdot)\}$ as public and keeps its master key mk_{RA} secret to it only.

6.4.2 Registration phase

In this phase, the trusted register authority (RA) initially registers all the IoT smart devices (SD_i) belonging to a particular gateway nodes (GW_j), the gateway nodes (GW_j), the edge servers (ES_l) and the cloud servers (CS_k) before their functioning in the IoT environment. The detailed descriptions of all the registration processes are given in the following subsections.

1) Gateway node registration phase

The RA enrols each gateway node GW_j before secure communication with its designated smart IoT devices for accessing the real time data directly. The following steps are involved in this phase:

- Step GWR_1: The RA first picks a unique identity ID_{GW_j}, a master secret key mk_{GW_j}, and a random secret (private) key $k_{GW_j} \in Z_q^*$ for each GW_j, and computes the public key corresponding to the secret key k_{GW_j} as $Pub_{GW_j} = k_{GW_j} \cdot G$.

- Step GWR_2: The RA then calculates a secret temporal credential as $TC_{GW_j} = h(ID_{GW_j} ||mk_{GW_j} ||RTS_{GW_j} ||mk_{RA})$, where RTS_{GW_j} is the registration timestamp of GW_j.

- Step GWR_3: After that then RA picks a unique temporary identity TID_{GW_j} corresponding to the real identity ID_{GW_j} and computes the pseudo-identity as $PID_{GW_j} = h(ID_{GW_j} ||mk_{GW_j})$.

- Step GWR_4: For each GW_j, the RA selects a distinct t_2-degree symmetric bivariate polynomial $g_j(x, y) = \sum_{m=0}^{t_2} \sum_{n=0}^{t_2} b_{m,n} x^m y^n$ over the Galois (finite) field $GF(q)$, where $b_{m,n} \in Z_q$ such that $t_2 \gg$ number of deployed gateway nodes GW_j in the IoT environment so that the proposed scheme becomes "t-collusion resistant and unconditionally secure" [57] and $g_j(x, y) = g_j(y, x)$. The RA also computes the polynomial share $g_j(PID_{GW_j}, y)$ for each GW_j, and makes Pub_{GW_j} as public.

After successful registration of each GW_j by the RA, the credentials stored in each GW_j's memory are shown in Figure 6.2.

$$TID_{GW_j}, PID_{GW_j}, TC_{GW_j}, (k_{GW_j}, Pub_{GW_j}), \{(TID_{SD_i}, PID_{SD_i}) | i = 1, 2, \cdots, n_{sd_j}\}, g_j(PID_{GW_j}, y), E_q(u, v), G, h(\cdot)$$

Figure 6.2: Pre-loaded credentials in GW_j's memory

2) IoT smart device registration phase

The RA registers each IoT smart device (SD_i) associated with a gateway node (GW_j) in the IoT environment by the following steps:

- Step SDR_1: The RA chooses a master key mk_{SD_i} for each SD_i associated with a GW_j in a particular application field of IoT (e.g., e-healthcare or smart home). The RA also picks a unique identity ID_{SD_i} and a random private key $k_{SD_i} \in Z_q^*$ for each SD_i, and calculates the corresponding public key $Pub_{SD_i} = k_{SD_i} \cdot G$.

- Step SDR_2: After computing Pub_{SD_i} in Step SDR_1, the RA calculates SD_i's temporal credential as $TC_{SD_i} = h(ID_{SD_i} ||mk_{SD_i} ||mk_{RA} ||mk_{GW_j} ||RTS_{SD_i})$, where RTS_{SD_i} is the registration timestamp of SD_i and mk_{GW_j} is the master key already selected for GW_j by the RA in Section 6.4.2.

- Step SDR_3: Next, the RA selects a unique temporary identity TID_{SD_i} corresponding to the real identity ID_{SD_i} for each SD_i and calculates its pseudo-identity as $PID_{SD_i} = h(ID_{SD_i} || mk_{SD_i})$, and makes Pub_{SD_i} as public.

After successful registration of each SD_i, the credentials stored in each SD_i's memory are shown in Figure 6.3.

$$TID_{SD_i}, PID_{SD_i}, TC_{SD_i}, (k_{SD_i}, Pub_{SD_i}),\ E_q(u,v),\ G,\ h(\cdot)$$

Figure 6.3: Pre-loaded credentials in SD_i's memory

3) Edge server registration phase

The RA registers all the edge servers ES_l, $(l = 1, 2, \cdots, n_{es})$, before secure communication with their associated gateway nodes. The following steps are executed in this phase:

- Step ESR_1: For each ES_l, the RA selects a unique identity ID_{ES_l}, a master secret key mk_{ES_l} and a random secret key $k_{ES_l} \in E_q(u,v)$, and computes the public key corresponding to the secret key k_{ES_l} as $Pub_{ES_l} = k_{ES_l} \cdot G$.

- Step ESR_2: For each ES_l, the RA also computes a secret temporal credential as $TC_{ES_l} = h(ID_{ES_l} || mk_{ES_l} || RTS_{ES_l} || mk_{RA})$, where RTS_{ES_l} is the registration timestamp of ES_l.

- Step ESR_3: Next, the RA picks a unique temporary identity TID_{ES_l} corresponding to the real identity ID_{ES_l} of each ES_l and computes its pseudo-identity as $PID_{ES_l} = h(ID_{ES_l} || mk_{ES_l})$.

- Step ESR_4: For each ES_l, the RA then takes the same selected t_2-degree symmetric bivariate polynomial as done in Step GWR_4 (see Section 6.4.2), where $g_j(x,y) = \sum_{m=0}^{t_2} \sum_{n=0}^{t_2} b_{m,n} x^m y^n$ over $GF(q)$, $b_{m,n} \in Z_q$, and another t_1-degree bivariate symmetric polynomial $f(x,y) = \sum_{i=0}^{t_1} \sum_{j=0}^{t_1} a_{i,j} x^i y^j$ over $GF(q)$ where $a_{i,j} \in Z_q$ such that $t_1 \gg$ number of deployed edge servers ES_l in the network so that the proposed scheme also becomes "t-collusion resistant and unconditionally secure" [57] and $f(x,y) = f(y,x)$. The RA computes the polynomial shares $g_j(PID_{ES_l}, y)$ and $f(PID_{ES_l}, y)$ for each ES_l, and makes Pub_{ES_l} as public.

After successful registration of each ES_l by the RA, the credentials stored in each ES_l's memory are shown in Figure 6.4.

$$TID_{ES_l}, PID_{ES_l}, TC_{ES_l}, (k_{ES_l}, Pub_{ES_l}), \{(TID_{GW_j}, PID_{GW_j}) | j = 1, 2, \cdots, n_{gw_l}\},$$
$$g_j(PID_{ES_l}, y), f(PID_{ES_l}, y), E_q(u, v), G, h(\cdot)$$

Figure 6.4: Pre-loaded credentials in ES_l's database

4) Cloud server registration phase

The RA requires to register all the cloud servers CS_k, $(k = 1, 2, \cdots, n_{cs})$, containing in the blockchain center (BC), prior to their secure communication with the associated edge servers. This phase requires execution of the following steps:

- Step CSR_1: For each CS_k, the RA needs to pick a unique identity ID_{CS_k}, a master secret key mk_{CS_k} and a random secret key $k_{CS_k} \in Z_q^*$, and then to compute the public key corresponding to the secret key k_{CS_k} as $Pub_{CS_k} = k_{CS_k} \cdot G$.
- Step CSR_2: The RA also selects a unique temporary identity TID_{CS_k} corresponding to the real identity ID_{CS_k}, and computes the pseudo-identity as $PID_{CS_k} = h(ID_{CS_k} || mk_{CS_k})$.
- Step CSR_3: For each CS_k, the RA selects the same t_1-degree symmetric bivariate polynomial as already done in Step ESR_4 (see Section 6.4.2): $f(x, y) = \sum_{i=0}^{t_1} \sum_{j=0}^{t_1} a_{i,j} x^i y^j$ where $a_{i,j} \in Z_q$ such that $t_1 \gg$ number of deployed cloud server CS_k in the cloud server network/blockchain center. Next, the RA calculates the polynomial share $f(PID_{CS_k}, y)$ for each CS_k, $k = 1, 2, \cdots, n_{cs}$, and makes Pub_{CS_k} as public.

After successful registration of each CS_k by the RA, the credentials pre-loaded in each CS_k's data are illustrated in Figure 6.5.

$$(k_{CS_k}, Pub_{CS_k}), \{(TID_{ES_l}, PID_{ES_l}) | l = 1, 2, \cdots, n_{es}\}, \{(TID_{CS_k}, PID_{CS_k}) | k = 1, 2, \cdots, n_{cs}\},$$
$$f(PID_{CS_k}, y), E_q(u, v), G, h(\cdot)$$

Figure 6.5: Pre-loaded credentials in CS_k's database

Remark 6.1. *The purpose of selecting two types of bivariate polynomials, such as* $f(x, y) = \sum_{i=0}^{t_1} \sum_{j=0}^{t_1} a_{i,j} x^i y^j$ *of degree* t_1 *and* $g_j(x, y) = \sum_{m=0}^{t_2} \sum_{n=0}^{t_2} b_{m,n} x^m y^n$ *of degree* t_2 *over the Galois field* $GF(q)$ *is as follows. The polynomial* $g_j(x, y)$ *has been used for establishing a session key between a registered gateway node and its associated edge server during the access control process in a particular session (see Section 6.4.3). On the other side, the polynomial* $f(x, y)$ *has been applied for establishing a pairwise secret key between an edge server and a cloud server during the key management procedure (see Section 6.4.4).*

6.4.3 Access control phase

In this phase, we discuss two types of access control: a) access control between a smart device and its gateway node, and b) access control between a gateway node and its associated edge server.

1) Access control between smart device and gateway node

In this phase, a registered smart device, say SD_i wants to validate itself with its associated gateway node, say GW_j in a particular application. Once that is done, SD_i will establish a session key for a particular session with its associated GW_j for secure communication. For this purpose, the following steps need to be executed:

- Step $ACSG_1$: SD_i first generates a random number $r_{SD_i} \in Z_q^*$, picks current timestamp TS_{SD_i}, and calculates $R_{SD_i} = h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) \cdot G$. After that SD_i generates a signature on r_{SD_i} as $Sign_i = h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) + h(Pub_{SD_i} \,||Pub_{GW_j} \,||TS_{SD_i} \,||TID_{SD_i} \,||PID_{SD_i}) * k_{SD_i} \pmod q$. Next, SD_i constructs a message $Msg_{SG_1} = \{TID_{SD_i},\ R_{SD_i},\ Sign_i,\ TS_{SD_i}\}$, and then sends it to the GW_j via open channel.

- Step $ACSG_2$: After receiving the message Msg_{SG_1} from SD_i at time TS'_{SD_i}, GW_j first verifies if $|TS'_{SD_i} - TS_{SD_i}| < \Delta T$ or not, where ΔT is the "maximum transmission delay". If the condition is not true, GW_j ignores the message. Otherwise, GW_j fetches PID_{SD_i} corresponding to TID_{SD_i} from its database, and calculates $h(Pub_{SD_i} \,||Pub_{GW_j} \,||TS_{SD_i} \,||TID_{SD_i} \,||PID_{SD_i})$. GW_j also verifies the signature of SD_i as $Sign_i.G \stackrel{?}{=} R_{SD_i} + h(Pub_{SD_i} \,||Pub_{GW_j} \,||TS_{SD_i} \,||TID_{SD_i} \,||PID_{SD_i}) \cdot Pub_{SD_i}$. If the signature is verified successfully, SD_i is believed to be a legitimate node and the next step will be executed; else, SD_i's request is rejected by GW_j.

- Step $ACSG_3$: GW_j generates a current timestamp TS_{GW_j} and a random number $r_{GW_j} \in Z_q^*$, and computes the corresponding public value $R_{GW_j} = h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}) \cdot G$. After that GW_j calculates the session key shared with SD_i as $SK_{GW_j,SD_i} = h(h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}).R_{SD_i} \,||PID_{SD_i})$. GW_j also generates a new unique temporary identity $TID_{SD_i}^{new}$ for SD_i and calculates $TID_{SD_i}^* = TID_{SD_i}^{new} \oplus h(SK_{GW_j,SD_i} \,||TID_{SD_i} \,||TS_{GW_j})$.

- Step $ACSG_4$: Now, GW_j generates a signature on r_{GW_j} as well as SK_{GW_j,SD_i} as $Sign_j = h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}) + h(TID_{SD_i} \,||PID_{SD_i} \,||Pub_{SD_i}$

$||Pub_{GW_j}||SK_{GW_j,SD_i}||TS_{GW_j}) * k_{GW_j} \pmod q$ and constructs a message $Msg_{SG_2} = \{TID^*_{SD_i}, R_{GW_j}, Sign_j, TS_{GW_j}\}$. After that GW_j dispatches the message Msg_{SG_2} to SD_i via public channel.

- Step $ACSG_5$: After receiving the message Msg_{SG_2} from GW_j at time TS'_{GW_j}, SD_i validates if $|TS'_{GW_j} - TS_{GW_j}| < \Delta T$ or not. If it is valid, SD_i calculates the session key shared with GW_j as $SK_{SD_i,GW_j} = h(h(r_{SD_i}||TC_{SD_i}||PID_{SD_i}||TS_{SD_i}).R_{GW_j}||PID_{SD_i})$. Next, SD_i verifies the signature of GW_j as $Sign_j \cdot G \stackrel{?}{=} R_{GW_j} + h(TID_{SD_i}||PID_{SD_i}||Pub_{SD_i}||Pub_{GW_j}||SK_{SD_i,GW_j}||TS_{GW_j}) \cdot Pub_{GW_j}$. If the signature is verified successfully, the GW_j is also trusted as a legitimate node by SD_i and the next step is executed; else, SD_i discards the GW_j's message Msg_{SG_2}.

- Step $ACSG_6$: SD_i extracts $TID^{new}_{SD_i}$ as $TID^{new}_{SD_i} = TID^*_{SD_i} \oplus h(SK_{SD_i,GW_j}||TID_{SD_i}||TS_{GW_j})$. Moreover, SD_i picks a current timestamp $TS^*_{SD_i}$ and calculates the session key verifier as $SKV_{SD_i,GW_j} = h(SK_{SD_i,GW_j}||TS^*_{SD_i})$ to build the last message $Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS^*_{SD_i}\}$. SD_i then dispatches Msg_{SG_3} to GW_j via open channel. At the same time, SD_i also updates TID_{SD_i} with the newly computed $TID^{new}_{SD_i}$ in its memory.

- Step $ACSG_7$: After receiving the message Msg_{SG_3} from SD_i at time $TS^{**}_{SD_i}$, GW_j validates if $|TS^{**}_{SD_i} - TS^*_{SD_i}| < \Delta T$ or not. If it holds, GW_j further verifies $SKV_{SD_i,GW_j} \stackrel{?}{=} h(SK_{GW_j,SD_i}||TS^*_{SD_i})$. GW_j also updates TID_{SD_i} with $TID^{new}_{SD_i}$ corresponding to PID_{SD_i} in its database in order to sync with SD_i.

Finally, both SD_i and GW_j will enjoy secure communication with the help of the established session key SK_{SD_i,GW_j} $(= SK_{GW_j,SD_i})$. The entire process is briefed in Figure 6.6.

2) Access control between gateway node and edge server

In this phase, the access control process between a registered gateway node, say GW_j and its associated edge server ES_l is discussed. After successful mutual authentication among GW_j and ES_l, both will agree on a session key for a particular session for secure communication. The following stages are involved:

- Step $ACGE_1$: GW_j generates a current timestamp TS_{GW_j} and a random number $r_{GW_j} \in Z^*_q$ to calculate the corresponding public value $R_{GW_j} = h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j}) \cdot G$ using its own private key k_{GW_j}. After that GW_j creates a signature on r_{GW_j} as $Sign_j = h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j})$

Smart device (SD_i)	Gateway Node (GW_j)
Generate random secret $r_{SD_i} \in Z_q^*$, current timestamp TS_{SD_i}.	
Compute $R_{SD_i} = h(r_{SD_i}\|\|TC_{SD_i}\|\|PID_{SD_i}\|\|TS_{SD_i}) \cdot G$,	
signature on r_{SD_i} as	
$Sign_i = h(r_{SD_i}\|\|TC_{SD_i}\|\|PID_{SD_i}\|\|TS_{SD_i})$	
$+ h(Pub_{SD_i}\|\|Pub_{GW_j}\|\|TS_{SD_i}\|\|TID_{SD_i}\|\|PID_{SD_i}) * k_{SD_i} \pmod q$.	
$\underrightarrow{Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}}$	
(via open channel)	Check $\|TS'_{SD_i} - TS_{SD_i}\| < \Delta T$? Accept/Reject?
	If so, fetch PID_{SD_i} corresponding to TID_{SD_i}.
	Compute $h(Pub_{SD_i}\|\|Pub_{GW_j}\|\|TS_{SD_i}\|\|TID_{SD_i}\|\|PID_{SD_i})$.
	Verify the signature
	$Sign_i.G \stackrel{?}{=} R_{SD_i} + h(Pub_{SD_i}\|\|Pub_{GW_j}\|\|TS_{SD_i}\|\|TID_{SD_i}\|\|PID_{SD_i}).Pub_{SD_i}$
	If so, generate random secret $r_{GW_j} \in Z_q^*$, current timestamp TS_{GW_j}.
	Compute $R_{GW_j} = h(r_{GW_j}\|\|TC_{GW_j}\|\|TID_{SD_i}\|\|PID_{SD_i}\|\|PID_{GW_j}\|\|TS_{GW_j}) \cdot G$,
	session key $SK_{GW_j,SD_i} = h(h(r_{GW_j}\|\|TC_{GW_j}\|\|TID_{SD_i}$
	$\|\|PID_{SD_i}\|\|PID_{GW_j}\|\|TS_{GW_j}).R_{SD_i}\|\|PID_{SD_i})$.
	Generate a new temporary identity $TID_{SD_i}^{new}$ for SD_i.
	Compute $TID_{SD_i}^* = TID_{SD_i}^{new} \oplus h(SK_{GW_j,SD_i}\|\|TID_{SD_i}\|\|TS_{GW_j})$.
	Compute signature on r_{GW_j} and SK_{GW_j,SD_i} as
	$Sign_j = h(r_{GW_j}\|\|TC_{GW_j}\|\|TID_{SD_i}\|\|PID_{SD_i}\|\|PID_{GW_j}\|\|TS_{GW_j}) + h(TID_{SD_i}$
	$\|\|PID_{SD_i}\|\|Pub_{SD_i}\|\|Pub_{GW_j}\|\|SK_{GW_j,SD_i}\|\|TS_{GW_j}) * k_{GW_j} \pmod q$.
	$\overleftarrow{Msg_{SG_2} = \{TID_{SD_i}^*, R_{GW_j}, Sign_j, TS_{GW_j}\}}$
	(via open channel)
Check $\|TS'_{GW_j} - TS_{GW_j}\| < \Delta T$? Accept/Reject?	
Compute $SK_{SD_i,GW_j} = h(h(r_{SD_i}\|\|TC_{SD_i}$	
$\|\|PID_{SD_i}\|\|TS_{SD_i}).R_{GW_j}\|\|PID_{SD_i})$.	
Verify signature $Sign_j \cdot G \stackrel{?}{=} R_{GW_j} + h(TID_{SD_i}\|\|PID_{SD_i}\|\|Pub_{SD_i}$	
$\|\|Pub_{GW_j}\|\|SK_{SD_i,GW_j}\|\|TS_{GW_j}).Pub_{GW_j}$	
Extract $TID_{SD_i}^{new} = TID_{SD_i}^* \oplus h(SK_{SD_i,GW_j}\|\|TID_{SD_i}\|\|TS_{GW_j})$.	
Generate a current timestamp $TS_{SD_i}^*$.	
Compute session key verifier: $SKV_{SD_i,GW_j} = h(SK_{SD_i,GW_j}\|\|TS_{SD_i}^*)$.	
Update TID_{SD_i} with $TID_{SD_i}^{new}$ corresponding to PID_{SD_i}.	
$\underrightarrow{Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS_{SD_i}^*\}}$	
(via open channel)	
	Verify $\|TS_{SD_i}^{**} - TS_{SD_i}^*\| < \Delta T$? Accept/Reject?
	Verify $SKV_{SD_i,GW_j} \stackrel{?}{=} h(SK_{GW_j,SD_i}\|\|TS_{SD_i}^*)$
	Update TID_{SD_i} with $TID_{SD_i}^{new}$ corresponding to PID_{SD_i}.
Both SD_i and GW_j store the shared common session key SK_{GW_j,SD_i} $(= SK_{SD_i,GW_j})$.	

Figure 6.6: Summary of access control between smart device (SD_i) and gateway node (GW_j)

$+h(TID_{GW_j}\ ||Pub_{GW_j}\ ||PID_{GW_j}\ ||TS_{GW_j})\ *k_{GW_j} \pmod q$. Next, GW_j constructs a message as $Msg_{GE_1} = \{TID_{GW_j}, R_{GW_j}, Sign_j, TS_{GW_j}\}$ and sends it to the edge server ES_l via public channel.

- Step $ACGE_2$: After receiving Msg_{GE_1} at time TS'_{GW_j} from GW_j, ES_l validates if $|TS'_{GW_j} - TS_{GW_j}| < \Delta T$ is satisfied. If it is verified successfully, ES_l fetches PID_{GW_j} corresponding to TID_{GW_j} from its database, and then checks the received signature by

the condition: $Sign_j.G \stackrel{?}{=} R_{GW_j} + h(TID_{GW_j} || Pub_{GW_j} || PID_{GW_j} || TS_{GW_j}) \cdot Pub_{GW_j}$. If the signature is verified successfully, GW_j is believed to be a legitimate entity to ES_l and ES_l executes the next step. Otherwise, ES_l rejects GW_j's Msg_{GE_1}.

- Step $ACGE_3$: ES_l now generates current timestamp TS_{ES_l} and random number $r_{ES_l} \in Z_q^*$, and calculates the respective public value $R_{ES_l} = h(PID_{GW_j} || TC_{ES_l} || PID_{ES_l} || r_{ES_l} || k_{ES_l} || TS_{ES_l}) \cdot G$. After that ES_l computes $g_j(PID_{ES_l}, PID_{GW_j})$ by evaluating its own polynomial share $g_j(PID_{ES_l}, y)$ at the point $y = PID_{GW_j}$, the Diffie-Hellman type key $DK_{lj} = h(PID_{GW_j} || TC_{ES_l} || PID_{ES_l} || r_{ES_l} || k_{ES_l} || TS_{ES_l}) \cdot R_{GW_j}$, the session key shared with GW_j as $SK_{ES_l,GW_j} = h(g_j(PID_{ES_l}, PID_{GW_j}) || DK_{lj})$ and $PID^*_{ES_l} = PID_{ES_l} \oplus h(PID_{GW_j} || Pub_{GW_j} || Pub_{ES_l} || TS_{ES_l})$. Furthermore, ES_l generates a signature on r_{ES_l} and SK_{ES_l,GW_j} as $Sign_l = h(PID_{GW_j} || TC_{ES_l} || PID_{ES_l} || r_{ES_l} || k_{ES_l} || TS_{ES_l}) + h(SK_{ES_l,GW_j} || PID_{GW_j} || Pub_{GW_j} || Pub_{ES_l}) * k_{ES_l} \pmod q$. In addition, ES_l generates a new temporary identity $TID^{new}_{GW_j}$ for GW_j and calculates $TID^*_{GW_j} = TID^{new}_{GW_j} \oplus h(TID_{GW_j} || SK_{ES_l,GW_j} || TS_{ES_l}) \pmod q$. After that, ES_l forms a message $Msg_{GE_2} = \{PID^*_{ES_l}, TID^*_{GW_j}, R_{ES_l}, Sign_l, TS_{ES_l}\}$ and sends it to GW_j via public channel.

- Step $ACGE_4$: After receiving the message Msg_{GE_2} at time TS'_{ES_l} from ES_l, GW_j checks the condition: $|TS'_{ES_l} - TS_{ES_l}| < \Delta T$. If it is not valid, GW_j rejects the message Msg_{GE_2}. Otherwise, GW_j derives $PID_{ES_l} = PID^*_{ES_l} \oplus h(PID_{GW_j} || Pub_{GW_j} || Pub_{ES_l} || TS_{ES_l})$ and computes $g_j(PID_{GW_j}, PID_{ES_l})$ by evaluating its own polynomial share $g_j(PID_{GW_j}, y)$ at the point $y = PID_{ES_l}$, the Diffie-Hellman type key $DK_{jl} = h(TID_{GW_j} || TC_{GW_j} || PID_{GW_j} || r_{GW_j} || k_{GW_j} || TS_{GW_j}) \cdot R_{ES_l}$ and the session key shared with ES_l as $SK_{GW_j,ES_l} = h(g_j(PID_{GW_j}, PID_{ES_l}) || DK_{jl})$. If the signature verification by the condition: $Sign_l \cdot G \stackrel{?}{=} R_{ES_l} + h(SK_{GW_j,ES_l} || PID_{GW_j} || Pub_{GW_j} || PID_{ES_l}) \cdot Pub_{ES_l}$ is successful, ES_l is believed to be legitimate principal to GW_j; otherwise, the session is terminated. Next, GW_j extracts $TID^{new}_{GW_j}$ as $TID^{new}_{GW_j} = TID^*_{GW_j} \oplus h(TID_{GW_j} || SK_{GW_j,ES_l} || TS_{ES_l}) \pmod q$. After that GW_j generates current timestamp $TS^*_{GW_j}$ and calculates the session key verifier as $SKV_{GW_j,ES_l} = h(SK_{GW_j,ES_l} || TS^*_{GW_j})$ in order to create a message $Msg_{GE_3} = \{SKV_{GW_j,ES_l}, TS^*_{GW_j}\}$. Finally, the message Msg_{GE_3} is dispatched by GW_j to ES_l via public channel. At the same time, GW_j also updates TID_{GW_j} with the newly computed $TID^{new}_{GW_j}$ in its database.

- Step $ACGE_5$: After receiving the message Msg_{GE_3} from GW_j at time $TS^{**}_{GW_j}$, ES_l validates the received timestamp $TS^*_{GW_j}$ by the condition: $|TS^{**}_{GW_j} - TS^*_{GW_j}| < \Delta T$. If

it is not valid, ES_l immediately rejects GW_j's message Msg_{GE_3}. On the other side, ES_l verifies the session key by the condition: $SKV_{GW_j,ES_l} \stackrel{?}{=} h(SK_{ES_l,GW_j} ||TS^*_{GW_j})$. If the signature is valid, ES_l updates the TID_{GW_j} with $TID^{new}_{GW_j}$ corresponding to PID_{GW_j} in its database.

At the end of this phase, both GW_j and ES_l share the same secret session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$. The entire method is summarized in Figure 6.7.

6.4.4 Key management phase between edge server and cloud server

In this phase, the key management procedure between an edge server (ES_l) and a cloud server (CS_k) in the blockchain center is explained. At the end of successful key management, both ES_l and CS_k will establish a secret key among them for secret communications.

The following steps are essential to fulfill this task:

- Step $KMEC_1$: ES_l first generates the current timestamp TS_{ES_l} and a random number $r_{ES_l} \in Z^*_q$ to calculate $f(PID_{ES_l}, PID_{CS_k})$ by evaluating its own polynomial share $f(PID_{ES_l}, y)$ using PID_{CS_k} of the CS_k, and $A_{ES_l} = h(r_{ES_l} ||PID_{ES_l} ||PID_{CS_k} ||TS_{ES_l} ||k_{ES_l}) \oplus h(f(PID_{ES_l}, PID_{CS_k}) ||TS_{ES_l})$. After that ES_l dispatches the message $Msg_{EC_1} = \{TID_{ES_l}, A_{ES_l}, TS_{ES_l}\}$ to CS_k via a public channel.

- Step $KMEC_2$: After receiving Msg_{EC_1} at time $TS^*_{ES_l}$, CS_k checks the validity of timestamp TS_{ES_l} by the condition: $|TS^*_{ES_l} - TS_{ES_l}| < \Delta T$. If the timestamp validation passes, it accepts the message Msg_{EC_1}; else, it rejects Msg_{EC_1}. CS_k then fetches PID_{ES_l} corresponding to TID_{ES_l} received in Msg_{EC_1}. Next, it calculates $h(r_{ES_l} ||PID_{ES_l} ||PID_{CS_k} ||TS_{ES_l} ||k_{ES_l}) = A_{ES_l} \oplus h(f(PID_{CS_k}, PID_{ES_l}) ||TS_{ES_l})$. Furthermore, it generates the current timestamp TS_{CS_k} and a random number $r_{CS_k} \in Z^*_q$ to calculate $B_{CS_k} = h(r_{CS_k} ||TS_{CS_k} ||k_{CS_k}) \oplus h(f(PID_{CS_k}, PID_{ES_l}) ||TS_{CS_k})$ and the secret key shared with ES_l as $SK_{CS_k,ES_l} = h(h(r_{ES_l} ||PID_{ES_l} ||PID_{CS_k} ||TS_{ES_l} ||k_{ES_l}) ||h(r_{CS_k} ||TS_{CS_k} ||k_{CS_k}) ||f(PID_{CS_k}, PID_{ES_l}))$. After computing the secret key, it generates a new temporary identity $TID^{new}_{ES_l}$ for ES_l, and calculates the secret key verifier $SKV_{CS_k,ES_l} = h(SK_{CS_k,ES_l} ||TS_{CS_k} ||TID^{new}_{ES_l})$, $C_{CS_k} = TID^{new}_{ES_l} \oplus h(TID_{ES_l} ||SK_{CS_k,ES_l} ||TS_{CS_k})$, $D_{CS_k} = PID_{CS_k} \oplus h(PID_{ES_l} ||TID_{ES_l} ||TS_{CS_k})$, and also updates TID_{ES_l} with $TID^{new}_{ES_l}$ corresponding to PID_{ES_l} in its database. Finally, CS_k sends the message $Msg_{EC_2} = \{B_{CS_k}, C_{CS_k}, D_{CS_k}, SKV_{CS_k,ES_l}, TS_{CS_k}\}$ to ES_l via open channel.

Gateway Node (GW_j)	Edge Server (ES_l)
Generate random number $r_{GW_j} \in Z_q^*$, current timestamp TS_{GW_j}. Compute public value $R_{GW_j} = h(TID_{GW_j}\|\|TC_{GW_j}\|\|PID_{GW_j}\|\|r_{GW_j}\|\|k_{GW_j}\|\|TS_{GW_j}) \cdot G$. Generate a signature on r_{GW_j} as $Sign_j = h(TID_{GW_j}\|\|TC_{GW_j}\|\|PID_{GW_j}\|\|r_{GW_j}\|\|k_{GW_j}\|\|TS_{GW_j})$ $+ h(TID_{GW_j}\|\|Pub_{GW_j}\|\|PID_{GW_j}\|\|TS_{GW_j}) * k_{GW_j} \pmod q$. $Msg_{GE_1} = \{TID_{GW_j}, R_{GW_j}, Sign_j, TS_{GW_j}\}$ $\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$ (via public channel)	Check $\|TS'_{GW_j} - TS_{GW_j}\| < \Delta T$? Accept/Reject? Fetch PID_{GW_j} corresponding to TID_{GW_j} and verify the signature: $Sign_j.G \stackrel{?}{=} R_{GW_j} + h(TID_{GW_j}\|\|Pub_{GW_j}\|\|PID_{GW_j}\|\|TS_{GW_j}).Pub_{GW_j}$ Generate current timestamp TS_{ES_l} and random number $r_{ES_l} \in Z_q^*$. Compute $R_{ES_l} = h(PID_{GW_j}\|\|TC_{ES_l}\|\|PID_{ES_l}\|\|r_{ES_l}\|\|k_{ES_l}\|\|TS_{ES_l}) \cdot G$. Compute $g_j(PID_{ES_l}, PID_{GW_j})$, $DK_{lj} = h(PID_{GW_j}\|\|TC_{ES_l}\|\|PID_{ES_l}\|\|r_{ES_l}\|\|k_{ES_l}\|\|TS_{ES_l}) \cdot R_{GW_j}$. Compute session key $SK_{ES_l,GW_j} = h(g_j(PID_{ES_l}, PID_{GW_j})\|\|DK_{lj})$, $PID^*_{ES_l} = PID_{ES_l} \oplus h(PID_{GW_j}\|\|Pub_{GW_j}\|\|Pub_{ES_l}\|\|TS_{ES_l})$, signature on r_{ES_l} and SK_{ES_l,GW_j} as $Sign_l = h(PID_{GW_j}\|\|TC_{ES_l}\|\|PID_{ES_l}\|\|r_{ES_l}\|\|k_{ES_l}\|\|TS_{ES_l})$ $+h(SK_{ES_l,GW_j}\|\|PID_{GW_j}\|\|Pub_{GW_j}\|\|Pub_{ES_l}) * k_{ES_l} \pmod q$. Generate new temporary identity $TID^{new}_{GW_j}$ for GW_j. Compute $TID^*_{GW_j} = TID^{new}_{GW_j} \oplus h(TID_{GW_j}\|\|SK_{ES_l,GW_j}\|\|TS_{ES_l}) \pmod q$. $Msg_{GE_2} = \{PID^*_{ES_l}, TID^*_{GW_j}, R_{ES_l}, Sign_l, TS_{ES_l}\}$ $\xleftarrow{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$ (via public channel)
Check if $\|TS'_{ES_l} - TS_{ES_l}\| < \Delta T$? Accept/Reject? Derive $PID_{ES_l} = PID^*_{ES_l} \oplus h(PID_{GW_j}\|\|Pub_{GW_j}\|\|Pub_{ES_l}\|\|TS_{ES_l})$. Compute $g_j(PID_{GW_j}, PID_{ES_l})$, $DK_{jl} = h(TID_{GW_j}\|\|TC_{GW_j}\|\|PID_{GW_j}\|\|r_{GW_j}\|\|k_{GW_j}\|\|TS_{GW_j}).R_{ES_l}$. Compute $SK_{GW_j,ES_l} = h(g_j(PID_{GW_j}, PID_{ES_l})\|\|DK_{jl})$. Verify signature $Sign_l.G \stackrel{?}{=} R_{ES_l} + h(SK_{GW_j,ES_l}\|\|PID_{GW_j}\|\|Pub_{GW_j}\|\|PID_{ES_l}).Pub_{ES_l}$. Derive $TID^{new}_{GW_j} = TID^*_{GW_j} \oplus h(TID_{GW_j}\|\|SK_{GW_j,ES_l}\|\|TS_{ES_l}) \pmod q$. Generate current timestamp $TS^*_{GW_j}$. Compute $SKV_{GW_j,ES_l} = h(SK_{GW_j,ES_l}\|\|TS^*_{GW_j})$. Update TID_{GW_j} with $TID^{new}_{GW_j}$ corresponding to PID_{GW_j}. $Msg_{GE_3} = \{SKV_{GW_j,ES_l}, TS^*_{GW_j}\}$ $\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$ (via public channel)	
	Verify $\|TS^{**}_{GW_j} - TS^*_{GW_j}\| < \Delta T$? Accept/Reject? Check $SKV_{GW_j,ES_l} \stackrel{?}{=} h(SK_{ES_l,GW_j}\|\|TS^*_{GW_j})$ If so, update TID_{GW_j} with $TID^{new}_{GW_j}$ corresponding to PID_{GW_j}.
Both GW_j and ES_l store the shared common session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$.	

Figure 6.7: Summary of access control between gateway node (GW_j) & edge server (ES_l)

- Step $KMEC_3$: After receiving Msg_{EC_2} from CS_k at time $TS^*_{CS_k}$, ES_l verifies the validity of the message by checking the timestamp TS_{CS_k} by the condition: $|TS^*_{CS_k} - TS_{CS_k}| < \Delta T$. If the condition is valid, ES_l computes $PID_{CS_k} = D_{CS_k} \oplus h(PID_{ES_l} ||TID_{ES_l} ||TS_{CS_k})$, $h(r_{CS_k} ||TS_{CS_k} ||k_{CS_k}) = B_{CS_k} \oplus h(f(PID_{ES_l}, PID_{CS_k}) ||TS_{CS_k})$,

and the secret key shared with CS_k as $SK_{ES_l,CS_k} = h(h(r_{ES_l} ||PID_{ES_l} ||PID_{CS_k} ||TS_{ES_l} ||k_{ES_l})|| h(r_{CS_k} ||TS_{CS_k} ||k_{CS_k}) ||f(PID_{ES_l}, PID_{CS_k}))$. In addition, ES_l calculates $TID^{new}_{ES_l} = C_{CS_k} \oplus h(TID_{ES_l} ||SK_{ES_l,CS_k} ||TS_{CS_k})$ and checks the validity of $SKV_{CS_k,ES_l} \stackrel{?}{=} h(SK_{ES_l,CS_k} ||TS_{CS_k} ||TID^{new}_{ES})$. If it is valid, CS_k is treated as authorized entity by ES_l, and accepts SK_{ES_l,CS_k} as the valid secret key. Furthermore, ES_l updates TID_{ES_l} with $TID^{new}_{ES_l}$ corresponding to PID_{ES_l} in its database.

The summary of this key management process is illustrated in Figure 6.8.

Edge Server (ES_l)	Cloud Server (CS_k)
Generate current timestamp TS_{ES_l} and random number $r_{ES_l} \in Z^*_q$.	
Compute $f(PID_{ES_l}, PID_{CS_k})$ using PID_{CS_k} of CS_k,	
$A_{ES_l} = h(r_{ES_l}\|\|PID_{ES_l}\|\|PID_{CS_k}\|\|TS_{ES_l}\|\|k_{ES_l}) \oplus h(f(PID_{ES_l}, PID_{CS_k})\|\|TS_{ES_l})$.	
$\underrightarrow{Msg_{EC_1} = \{TID_{ES_l}, A_{ES_l}, TS_{ES_l}\}}$	
(via open channel)	
	Check validity on timestamp TS_{ES_l}. Accept/Reject?
	If so, fetch PID_{ES_l} corresponding to TID_{ES_l}.
	Compute $h(r_{ES_l}\|\|PID_{ES_l}\|\|PID_{CS_k}\|\|TS_{ES_l}\|\|k_{ES_l})$
	$= A_{ES_l} \oplus h(f(PID_{CS_k}, PID_{ES_l})\|\|TS_{ES_l})$.
	Generate current timestamp TS_{CS_k} and random number $r_{CS_k} \in Z^*_q$.
	Compute $B_{CS_k} = h(r_{CS_k}\|\|TS_{CS_k}\|\|k_{CS_k}) \oplus h(f(PID_{CS_k}, PID_{ES_l})\|\|TS_{CS_k})$,
	and the secret key shared with ES_l as
	$SK_{CS_k,ES_l} = h(h(r_{ES_l}\|\|PID_{ES_l}\|\|PID_{CS_k}\|\|TS_{ES_l}\|\|k_{ES_l})$
	$\|\|h(r_{CS_k}\|\|TS_{CS_k}\|\|k_{CS_k})\|\|f(PID_{CS_k}, PID_{ES_l}))$.
	Generate new temporary identity $TID^{new}_{ES_l}$,
	secret key verifier $SKV_{CS_k,ES_l} = h(SK_{CS_k,ES_l}\|\|TS_{CS_k}\|\|TID^{new}_{ES_l})$.
	Compute $C_{CS_k} = TID^{new}_{ES_l} \oplus h(TID_{ES_l}\|\|SK_{CS_k,ES_l}\|\|TS_{CS_k})$,
	$D_{CS_k} = PID_{CS_k} \oplus h(PID_{ES_l}\|\|TID_{ES_l}\|\|TS_{CS_k})$.
	Update TID_{ES_l} with $TID^{new}_{ES_l}$ corresponding to PID_{ES_l}.
	$\overleftarrow{Msg_{EC_2} = \{B_{CS_k}, C_{CS_k}, D_{CS_k}, SKV_{CS_k,ES_l}, TS_{CS_k}\}}$
	(via open channel)
Check validity of timestamp TS_{CS_k}. Accept/Reject?	
If valid, derive $PID_{CS_k} = D_{CS_k} \oplus h(PID_{ES_l}\|\|TID_{ES_l}\|\|TS_{CS_k})$.	
Compute $h(r_{CS_k}\|\|TS_{CS_k}\|\|k_{CS_k}) = B_{CS_k} \oplus h(f(PID_{ES_l}, PID_{CS_k})\|\|TS_{CS_k})$,	
secret key $SK_{ES_l,CS_k} = h(h(r_{ES_l}\|\|PID_{ES_l}\|\|PID_{CS_k}\|\|TS_{ES_l}\|\|k_{ES_l})\|\|$	
$h(r_{CS_k}\|\|TS_{CS_k}\|\|k_{CS_k})\|\|f(PID_{ES_l}, PID_{CS_k}))$,	
$TID^{new}_{ES_l} = C_{CS_k} \oplus h(TID_{ES_l}\|\|SK_{ES_l,CS_k}\|\|TS_{CS_k})$.	
Check if $SKV_{CS_k,ES_l} = h(SK_{ES_l,CS_k}\|\|TS_{CS_k}\|\|TID^{new}_{ES})$?	
If valid, CS_k is treated as legitimate and accept SK_{ES_l,CS_k}.	
Update TID_{ES_l} with $TID^{new}_{ES_l}$ corresponding to PID_{ES_l}.	
Both ES_l and CS_k share $SK_{ES_l,CS_k} (= SK_{CS_k,ES_l})$ as a common secret key.	

Figure 6.8: Summary of key management phase between edge server (ES_l) and cloud server (CS_k)

6.4.5 Dynamic nodes addition phase

In this section, we discuss the mechanisms to dynamically add IoT smart devices as well as the gateway nodes. The IoT smart device deployment is extremely essential due to power

exhaustion of smart devices for their resource limitations or physical capturing of some IoT smart devices (as explained in our threat model in Section 6.3.2).

1) New IoT smart device addition

To deploy a new IoT smart device, say SD_{new} in a particular IoT application (for example, the j^{th} application associated with the gateway node GW_j), the registration authority RA executes the following steps:

- Step SDA_1: The RA picks a master key $mk_{SD_{new}}$ and a unique identity $ID_{SD_{new}}$ for SD_{new}. Next, the RA picks a random private key $k_{SD_{new}} \in Z_q^*$ and computes corresponding public key $Pub_{SD_{new}} = k_{SD_{new}} \cdot G$ for SD_{new}.
- Step SDA_2: The RA calculates SD_{new}'s temporal credential as $TC_{SD_{new}} = h(ID_{SD_{new}} ||mk_{SD_{new}} ||mk_{RA} ||mk_{GW_j} ||RTS_{SD_{new}})$, where $RTS_{SD_{new}}$ is the new registration timestamp of SD_{new} and mk_{GW_j} is the master key that was already stored in the RA database.
- Step SDA_3: Next, the RA selects a unique temporary identity $TID_{SD_{new}}$ corresponding to the real identity $ID_{SD_{new}}$ for SD_{new} and calculates SD_{new}'s pseudo-identity as $PID_{SD_{new}} = h(ID_{SD_{new}} ||mk_{SD_{new}})$, and makes $Pub_{SD_{new}}$ as public.
- Step SDA_4: Finally, the RA loads the secret information $\{TID_{SD_{new}}, PID_{SD_{new}}, TC_{SD_{new}}, (k_{SD_{new}}, Pub_{SD_{new}}), E_q(u, v), G, h(\cdot)\}$ into SD_{new}'s memory prior to its placement in the IoT application. Furthermore, the RA also sends the information $(TID_{SD_{new}}, PID_{SD_{new}})$ to the associated gateway node GW_j by secure channel.

The summary of a new IoT smart device (SD_{new}) addition phase is provided in Figure 6.9.

2) New gateway node addition

For adding a new gateway node, say GW_{new} in a particular IoT application with its designated smart IoT devices SD_i, the following steps are associated:

- Step GWA_1: For GW_{new}, the RA first picks a unique identity $ID_{GW_{new}}$, a master secret key $mk_{GW_{new}}$, and a random secret (private) key $k_{GW_{new}} \in Z_q^*$. The RA then computes the public key corresponding to the secret key $k_{GW_{new}}$ as $Pub_{GW_{new}} = k_{GW_{new}} \cdot G$.
- Step GWA_2: The RA calculates a secret temporal credential as $TC_{GW_{new}} = h(ID_{GW_{new}} ||mk_{GW_{new}} ||RTS_{GW_{new}} ||mk_{RA})$, where $RTS_{GW_{new}}$ is the new registration timestamp of GW_{new}.

Registration authority (RA)	New IoT smart device (SD_{new})
Pick master key $mk_{SD_{new}}$ and unique identity $ID_{SD_{new}}$ for SD_{new}. Select random private key $k_{SD_{new}} \in Z_q^*$ for SD_{new}. Calculate public key $Pub_{SD_{new}} = k_{SD_{new}} \cdot G$, temporal credential $TC_{SD_{new}} = h(ID_{SD_{new}} \|\|mk_{SD_{new}}$ $\|\|mk_{RA} \|\|mk_{GW_j} \|\|RTS_{SD_{new}})$ for SD_{new}. Pick unique temporary identity $TID_{SD_{new}}$ corresponding to $ID_{SD_{new}}$. Calculate pseudo-identity as $PID_{SD_{new}} = h(ID_{SD_{new}} \|\|mk_{SD_{new}})$. Make $Pub_{SD_{new}}$ as public.	Credentials: $\{TID_{SD_{new}}, PID_{SD_{new}}, TC_{SD_{new}}$, $(k_{SD_{new}}, Pub_{SD_{new}})$, $E_q(u,v)$, G, $h(\cdot)\}$ are loaded into its memory.

Figure 6.9: Summary of dynamic IoT smart device SD_{new} addition phase

- Step GWA_3: Next, the RA picks a unique temporary identity $TID_{GW_{new}}$ respective to the real identity $ID_{GW_{new}}$ and calculates the pseudo-identity $PID_{GW_{new}} = h(ID_{GW_{new}} \|\|mk_{GW_{new}})$.

- Step GWA_4: The RA computes the polynomial share $g_j(PID_{GW_{new}}, y)$ for GW_{new} that is used for secure communication with its associated edge server, say ES_l. The RA then makes $Pub_{GW_{new}}$ as public. Furthermore, the RA stores the credentials $\{TID_{GW_{new}}$, $PID_{GW_{new}}$, $TC_{GW_{new}}$, $(k_{GW_{new}}, Pub_{GW_{new}})$, $\{(TID_{SD_i}, PID_{SD_i})| \ i = 1, 2, \cdots, n_{sd_j}\}$, $g_j(PID_{GW_{new}}, y)$, $E_q(u,v)$, G, $h(\cdot)\}$ into GW_{new}'s database prior to its deployment. Finally, the RA sends the information $(TID_{GW_{new}}, PID_{GW_{new}})$ associated with of GW_{new} to the respective edge server ES_l by secure channel.

The summary of a new gateway node (GW_{new}) addition phase is also provided in Figure 6.10.

6.4.6 Blocks creation and addition in blockchain

This section provides the detailed procedures for creating, verifying and then adding a created block by an edge server ES_l in the P2P ES network.

1) Block creation

In the proposed CBACS-EIoT, the data are securely aggregated by a gateway node (GW_j) from their respective IoT smart devices (SD_i) using their established secret keys through

Registration authority (RA)	New gateway node (GW_{new})
Choose unique identity $ID_{GW_{new}}$, master secret key $mk_{GW_{new}}$, random secret (private) key $k_{GW_{new}} \in Z_q^*$ for GW_{new}. Calculate public key $Pub_{GW_{new}} = k_{GW_{new}} \cdot G$, secret temporal credential $TC_{GW_{new}} = h(ID_{GW_{new}} \|\| mk_{GW_{new}} \|\| RTS_{GW_{new}} \|\| mk_{RA})$. Choose unique temporary identity $TID_{GW_{new}}$ corresponding to $ID_{GW_{new}}$. Compute pseudo-identity $PID_{GW_{new}} = h(ID_{GW_{new}} \|\| mk_{GW_{new}})$, polynomial share $g_j(PID_{GW_{new}}, y)$ for GW_{new}. Make $Pub_{GW_{new}}$ as public.	Credentials: $\{TID_{GW_{new}}, PID_{GW_{new}}, TC_{GW_{new}}, (k_{GW_{new}}, Pub_{GW_{new}}), \{(TID_{SD_i}, PID_{SD_i}) \mid i = 1, 2, \cdots, n_{sd_j}\}, g_j(PID_{GW_{new}}, y), E_q(u, v), G, h(\cdot)\}$ are loaded into its memory.

Figure 6.10: Summary of dynamic gateway node GW_{new} addition phase

the access control phase described in Section 6.4.3. The gateway node GW_j then forms various transactions Tx_m $(m = 1, 2, \cdots, n_t)$, that are further used in block construction by its associated edge server ES_l. Using the shared secret key SK_{GW_j,ES_l} established during the access control phase between a gateway node and an edge server described in Section 6.4.3, GW_j sends all the transactions securely to ES_l. Now, based on the application type, there are three types of block formation done by the ES_l: a) formation of a block on public blockchain, b) formation of a block on private blockchain, and c) formation of a block on consortium blockchain as follows.

- In public blockchain, assume there is a block, say $Block_p$ containing block header and block payload. A block header in a public blockchain context contains block version (BV), previous block hash ($PBHash$), Merkle tree root (MTR), block type (public), creation time of block (timestamp, TS), owner of block (ES_l), and public key of signer (ES_l), that is, Pub_{ES_l}. In block payload, we have the n_t transactions $\{Tx_1, Tx_2, \cdots, Tx_{n_t}\}$. A Merkle tree is constructed on these transactions first, and then the Merkle Tree Toot (MTR) is calculated with the help of these transactions by ES_l. ES_l computes the hash of the current block, $CBHash$ as $CBHash = h(BV \|\| PBHash \|\| MTR \|\| Public \|\| TS \|\| ES_l \|\| Pub_{ES_l} \|\| Tx_1 \|\| Tx_2 \|\| \cdots \|\| Tx_{n_t})$. Next, ES_l calculates the signature $BSign$ on $CBHash$ using the signature generation of the "Elliptic Curve Digital Signature Algorithm (ECDSA)" [104].

- In private blockchain, the block header remains same as that for public block except the block type will be treated as "private". In this case, the transactions are encrypted using the ECC-based public key encryption $E(\cdot)$ with the help of the public key Pub_{ES_l} of ES_l. The transactions $\{E_{Pub_{ES_l}}(Tx_i) \ | i = 1, 2, \cdots, n_t\}$ are then used to calculate the Merkle tree root (MTR). ES_l computes the hash of the current block, $CBHash$ as $CBHash = h(BV || \ PBHash || \ MTR || \ Private || \ TS || \ ES_l || \ Pub_{ES_l} || \ E_{Pub_{ES_l}}(Tx_1) \ || E_{Pub_{ES_l}}(Tx_2) \ || \cdots || E_{Pub_{ES_l}}(Tx_{n_t}))$ and the signature $BSign$ on $CBHash$ using the ECDSA signature generation algorithm.

- In consortium blockchain, a block $Block_p$ containing the mixed types of transactions, where some transactions can be encrypted using the public key Pub_{ES_l} of ES_l and remaining transactions can be unencrypted, say $\{E_{Pub_{ES_l}}(Tx_1), Tx_2, \cdots, E_{Pub_{ES_l}}(Tx_{n_t})\}$. In this scenario, the block header remains same as that for public/private block except the block type will be treated as "hybrid". The transactions $\{E_{Pub_{ES_l}}(Tx_1), Tx_2, \cdots, E_{Pub_{ES_l}}(Tx_{n_t})\}$ are then applied to calculate the Merkle tree root (MTR). In addition, ES_l computes the hash of the current block, $CBHash$ as $CBHash = h(BV || \ PBHash || \ MTR || \ Hybrid || \ TS || \ ES_l || \ Pub_{ES_l} || \ E_{Pub_{ES_l}}(Tx_1) \ || Tx_2 \ || \cdots || E_{Pub_{ES_l}}(Tx_{n_t}))$ and the signature $BSign$ on $CBHash$ using the ECDSA signature generation algorithm.

Figures 6.11, 6.12 and 6.13 show the structures of various blocks in public, private, and consortium blockchains, respectively.

2) Block addition in blockchain

After constructing a $Block_p$ by ES_l, that block will be now added into the blockchain. To do so, the ES_l will first selects a leader, say L among the available n_{ES} edge servers acting as miner nodes in the P2P ES network using the existing leader selection algorithm suggested by Zhang *et al.* [239]. In this work, we consider two types of ledgers: a) local ledger that is maintained by each edge server in the P2P ES network, and b) global ledger that is maintained by the cloud servers in the blockchain center. The local ledger containing the blocks is securely sent by an edge server to the cloud servers for updating those blocks in the global ledger. This procedure is done periodically by the edge servers in order to avoid much burden at the cloud servers. We also assume that there will be synchronization among the edge servers so that the contents of the local ledgers maintained at the edge servers will be updated accordingly.

We apply the well-known "Ripple Protocol Consensus Algorithm (RPCA)" [209] for verification of a block $Block_p$ and addition of that block through the voting mechanism to the

Block Header	
Block Version	BV
Previous Block Hash	$PBHash$
Merkle Tree Root	MTR
Block Type	Public
Timestamp	TS
Owner of Block	ES_l
Public key of signer (ES_l)	Pub_{ES_l}
Block Payload (Transactions)	
Transaction #1 Transaction #2 ⋮ Transaction #n_t	Tx_1 Tx_2 ⋮ Tx_{n_t}
Current Block Hash	$CBHash$
Signature on block using ECDSA	$BSign$

Figure 6.11: Formation of a block on public blockchain

local ledgers of the edge servers. Note that each ES_l has a pair of private-public keys (k_{ES_l}, Pub_{ES_l}), where $Pub_{ES_l} = k_{ES_l} \cdot G$. Without any loss of generality, we assume that a block $Block_p$ has the content as $Block_p = \{BV, PBHash, MTR, Public, TS, ES_l, Pub_{ES_l}, \{Tx_i | i = 1, 2, \cdots, n_t\}, CBHash, BSign\}$ for public blockchain. Similarly, one can also consider other types of blocks in case of private and consortium blockchains. The procedure for verifying and adding the block $Block_p$ in the local ledgers of the edge servers is provided in Algorithm 4.

6.5 Security analysis

In this section, we first provide the correctness of the establishment of the secret keys during the access control and key management phases among different communicating entities. Next, we provide a rigorous security analysis using both the formal security under the standard random oracle model and the non-mathematical (informal) security analysis. Wang *et al.* [204] analyzed several two-factor based authentication protocols. They observed that under the accepted adversarial models, there are specific goals that are beyond attainment. They

Algorithm 4 Consensus protocol for block verification and addition in blockchain

Input: A block $Block_p = \{BV, PBHash, MTR, Public, TS, ES_l, Pub_{ES_l}, \{Tx_i \mid i = 1, 2, \cdots, n_t\}, CBHash, BSign\}$

Output: Verification and addition of block in the blockchain

1: The leader L creates timestamp TS_l and picks a cryptographic puzzle CPZ_l for each follower edge server node ES_l in the P2P ES network and starts voting process.
2: L encrypts the voting request (VTR) and puzzle using ECC-based public key cryptosystem with ES_l's public key Pub_{ES_l}, and sends $E_{Pub_{ES_l}}(CPZ_l, VTR, TS_l)$, the ECSDA signature on (CPZ_l, VTR, TS_l) using its own private key k_L, and $Block_p$ to each follower ES_l, ($l = 1, 2, \cdots, n_{es}$), with $ES_l \neq L$.
3: After receiving the message by each follower ES_l from leader L at time TS_l^*
4: **for** each peer node ES_l **do**
5: Calculate Merkle tree root (MTR^*) on transactions $\{Tx_i \mid i = 1, 2, \cdots, n_t\}$ in $Block_p$.
6: **if** ($MTR^* \neq MTR$) **then**
7: Terminate the consensus process.
8: **else**
9: Compute block hash $CBHash^*$ on $Block_p$ as $CBHash^* = h(BV || PBHash || MTR^* || Public || TS || ES_l || Pub_{ES_l} || Tx_1 || Tx_2 || \cdots || Tx_{n_t})$.
10: **if** ($CBHash^* = CBHash$) **then**
11: Verify the signature $BSign$ on $CBHash^*$ in received $Block_p$ using the ECDSA signature verification algorithm.
12: **if** signature is valid **then**
13: Decrypt the encrypted voting request with private key k_{ES_l} as $(CPZ_l^*, VTR^*, TS_l) = D_{k_{ES_l}}[E_{Pub_{ES_l}}(CPZ_l, VTR, TS_l)]$ using
14: the ECC decryption algorithm $D(\cdot)$.
15: **if** ($|TS_l^* - TS_l| < \Delta T$) **then**
16: Solve puzzle CPZ_l^* (set puzzle answer as SPZ_l) and verify block (set verification status as $BVStatus$).
17: Send the message containing verification status ($BVStatus$), solved puzzle answer (SPZ_l), and voting reply (VTL)
18: as $\{E_{Pub_L}((CPZ_l^*, SPZ_l), (VTR^*, VTL), BVStatus)\}$ to the leader L.
19: **end if**
20: **end if**
21: **end if**
22: **end if**
23: **end for**
24: L sets $VTCount \leftarrow 0$, where $VTCount$ is the valid vote counter.
L also sets $flag_{ES_l} = 0, \forall l = 1, 2, \cdots, n_{ES}, L \neq ES_l$.
25: **for** each received message from the responders ES_l **do**
26: L decrypts the message as $((CPZ_l^*, SPZ_l), (VTR^*, VTL), BVStatus) = D_{k_L}[E_{Pub_L}((CPZ_l^*, SPZ_l), (VTR^*, VTL), BVStatus)]$.
27: **if** (($CPZ_l^* = CPZ_l$) and ($VTR^* = VTR$)) **then**
28: **if** (($SPZ_l = valid$) and ($VTL = valid$) and ($BVStatus = valid$) and ($flag_{ES_l} = 0$)) **then**
29: L sets $VTCount = VTCount + 1$ and $flag_{ES_l} = 1$.
30: **end if**
31: **end if**
32: **end for**
33: **if** ($VTCount$ is more than 50% of the votes) **then**
34: Transaction enters the next round.
35: **if** ($VTCount$ less than the pre-defined threshold value, that is, 80% of the votes) **then**
36: go to Step 26.
37: **else**
38: Send the commit response to all other followers ES_l.
39: Add block $Block_p$ into the blockchain in the local ledgers and complete the consensus process.
40: **end if**
41: **end if**

Block Header	
Block Version	BV
Previous Block Hash	$PBHash$
Merkle Tree Root	MTR
Block Type	Private
Timestamp	TS
Owner of Block	ES_l
Public key of signer (ES_l)	Pub_{ES_l}
Block Payload (Encrypted Transactions)	
Encrypted Transaction #1 Encrypted Transaction #2 ⋮ Encrypted Transaction $\#n_t$	$E_{Pub_{ES_l}}(Tx_1)$ $E_{Pub_{ES_l}}(Tx_2)$ ⋮ $E_{Pub_{ES_l}}(Tx_{n_t})$
Current Block Hash	$CBHash$
Signature on block using ECDSA	$BSign$

Figure 6.12: Formation of a block on private blockchain

further pointed out that the formal methods based security analysis including random oracle based model may not capture some structural mistakes. Therefore, assuring the soundness of security protocols still continues as an open problem. Due to this important observations, we also consider the formal security verification using a widely-accepted automated software verification tool in Section 6.6.

6.5.1 Correctness proof

We show that the established secret keys among different communicating entities during the access control and key management phases are correct using the theorems 6.1, 6.2 and 6.3.

Theorem 6.1. *The session keys established between a smart device (SD_i) and its gateway node (GW_j) are the same during the access control phase in Section 6.4.3.*

Proof. During the access control phase, the gateway node GW_j generates the session key shared with the smart device SD_i as

$$SK_{GW_j,SD_i} = h(h(r_{GW_j}||TC_{GW_j}||TID_{SD_i}||PID_{SD_i}||PID_{GW_j}||TS_{GW_j}).R_{SD_i}||PID_{SD_i}), \quad (6.1)$$

Block Header	
Block Version	BV
Previous Block Hash	$PBHash$
Merkle Tree Root	MTR
Block Type	Hybrid
Timestamp	TS
Owner of Block	ES_l
Public key of signer (ES_l)	Pub_{ES_l}
Block Payload	
Encrypted Transaction #1 Transaction #2 ⋮ Encrypted Transaction #n_t	$E_{Pub_{ES_l}}(Tx_1)$ Tx_2 ⋮ $E_{Pub_{ES_l}}(Tx_{n_t})$
Current Block Hash	$CBHash$
Signature on block using ECDSA	$BSign$

Figure 6.13: Formation of a block on consortium blockchain

where $R_{SD_i} = h(r_{SD_i} ||TC_{SD_i} ||PID_{SD_i} ||TS_{SD_i}) \cdot G$. In a similar way, SD_i also generates the session key shared with the smart device GW_j as

$$SK_{SD_i,GW_j} = h(h(r_{SD_i}||TC_{SD_i}||PID_{SD_i}||TS_{SD_i}).R_{GW_j}||PID_{SD_i}), \tag{6.2}$$

where $R_{GW_j} = h(r_{GW_j} ||TC_{GW_j} ||TID_{SD_i} ||PID_{SD_i} ||PID_{GW_j} ||TS_{GW_j}) \cdot G$.

Now, we have,

$$\begin{aligned} & h(r_{GW_j}||TC_{GW_j}||TID_{SD_i} \\ ||PID_{SD_i}||PID_{GW_j}||TS_{GW_j}).R_{SD_i} &= (h(r_{GW_j}||TC_{GW_j}||TID_{SD_i}||PID_{SD_i}||PID_{GW_j}||TS_{GW_j}) \\ &\quad *h(r_{SD_i}||TC_{SD_i}||PID_{SD_i}||TS_{SD_i})) \cdot G \\ &= h(r_{SD_i}||TC_{SD_i}||PID_{SD_i}||TS_{SD_i}) \cdot \\ &\quad (h(r_{GW_j}||TC_{GW_j}||TID_{SD_i}||PID_{SD_i}||PID_{GW_j}||TS_{GW_j}) \cdot G) \\ &= h(r_{SD_i}||TC_{SD_i}||PID_{SD_i}||TS_{SD_i}) \cdot R_{GW_j} \end{aligned} \tag{6.3}$$

The result in Eq. (6.3) is used in both Eqs. (6.1) and (6.2) to have $SK_{GW_j,SD_i} = SK_{SD_i,GW_j}$. □

Theorem 6.2. *The session keys established between a gateway node GW_j and its associated edge server ES_l are the same during the access control phase in Section 6.4.3.*

Proof. During the access control phase, the session keys established by GW_j and ES_l shared with ES_l and GW_j are repectively

$$SK_{GW_j,ES_l} = h(g_j(PID_{GW_j}, PID_{ES_l})||DK_{jl}), \tag{6.4}$$

where $DK_{jl} = h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j}) \cdot R_{ES_l}$,

$$SK_{ES_l,GW_j} = h(g_j(PID_{ES_l}, PID_{GW_j})||DK_{lj}), \tag{6.5}$$

where $DK_{lj} = h(PID_{GW_j}||TC_{ES_l}||PID_{ES_l}||r_{ES_l}||k_{ES_l}||TS_{ES_l}) \cdot R_{GW_j}$. Now, we have the following derivation:

$$\begin{aligned} DK_{jl} &= h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j}) \cdot R_{ES_l} \\ &= (h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j}) \\ &\quad *h(PID_{GW_j}||TC_{ES_l}||PID_{ES_l}||r_{ES_l}||k_{ES_l}||TS_{ES_l})) \cdot G \\ &= h(PID_{GW_j}||TC_{ES_l}||PID_{ES_l}||r_{ES_l}||k_{ES_l}||TS_{ES_l}) \\ &\quad \cdot((h(TID_{GW_j}||TC_{GW_j}||PID_{GW_j}||r_{GW_j}||k_{GW_j}||TS_{GW_j}) \cdot G) \\ &= h(PID_{GW_j}||TC_{ES_l}||PID_{ES_l}||r_{ES_l}||k_{ES_l}||TS_{ES_l}) \cdot R_{GW_j} \\ &= DK_{lj}. \end{aligned} \tag{6.6}$$

It follows from Eqs. (6.4), (6.5) and (6.6) that

$$\begin{aligned} SK_{GW_j,ES_l} &= h(g_j(PID_{GW_j}, PID_{ES_l})||DK_{jl}) \\ &= h(g_j(PID_{ES_l}, PID_{GW_j})||DK_{jl}) \\ &= h(g_j(PID_{ES_l}, PID_{GW_j})||DK_{lj}) \\ &= SK_{ES_l,GW_j}, \end{aligned}$$

since $g_j(PID_{GW_j}, PID_{ES_l}) = g_j(PID_{ES_l}, PID_{GW_j})$. □

Theorem 6.3. *The secret keys established between an edge server (ES_l) and a cloud server (CS_k) are the same during the key management phase in Section 6.4.4.*

Proof. During the key management phase, ES_l generates a secret key shared with CS_k as

$$\begin{aligned} SK_{ES_l,CS_k} &= h(h(r_{ES_l}||PID_{ES_l}||PID_{CS_k}||TS_{ES_l}||k_{ES_l}) \\ &\quad ||h(r_{CS_k}||TS_{CS_k}||k_{CS_k})||f(PID_{ES_l}, PID_{CS_k})). \end{aligned} \tag{6.7}$$

Similarly, CS_k generates a secret key shared with ES_l as

$$\begin{aligned} SK_{CS_k,ES_l} &= h(h(r_{ES_l}||PID_{ES_l}||PID_{CS_k}||TS_{ES_l}||k_{ES_l}) \\ &\quad ||h(r_{CS_k}||TS_{CS_k}||k_{CS_k})||f(PID_{CS_k}, PID_{ES_l})). \end{aligned} \tag{6.8}$$

Since the chosen bivariate polynomial $f(x, y)$ is symmetric, we have $f(PID_{ES_l}, PID_{CS_k}) = f(PID_{CS_k}, PID_{ES_l})$. Hence, from Eqs. (6.7) and (6.8), it follows that $SK_{ES_l,CS_k} = SK_{CS_k,ES_l}$. □

6.5.2 Formal security analysis

To prove the security of the session keys established between a smart device (SD_i) and its gateway node (GW_j), between a gateway node (GW_j) and its associated edge server (ES_l), and also between an edge server (ES_l) and its associated cloud server (CS_k) against an adversary $\mathcal{A}$ in our proposed scheme (CBACS-EIoT), we apply the wide-accepted "Real-Or-Random (ROR) oracle model" [10]. Using the ROR model, we use the semantic security approach to prove the session key security of CBACS-EIoT in Theorem 6.4.

The ROR model is described by considering the following random oracles:

- **Participants:** At the time of the access control phase, the entities, namely smart devices (SD_i) and gateway nodes (GW_j) are engaged apart from the RA which is only involved during the registration and dynamic node addition phases. Here, $\Psi_{SD}^{l_1}$ and $\Psi_{GW}^{l_2}$ are used to represent the l_1^{th} and l_2^{th} instances of SD_i and GW_j, respectively. These instances are termed as the "random oracles".

- **Accepted state:** An instance Ψ^l will go to its "accepted state", when the last valid protocol message is accepted. All the communication messages can be ordered in sequentially, and these together form the "session identification *sid* of Ψ^l for the currently executed session".

- **Partnering:** Two instances (Ψ^{l_1} and Ψ^{l_2}) are assumed to be partners to each other, if they follow the following rules:

  - Ψ^{l_1} and Ψ^{l_2} require to be in "accepted states".
  - Ψ^{l_1} and Ψ^{l_2} require to share the same *sid*.
  - Ψ^{l_1} and Ψ^{l_2} need to be "mutual partners of each other".

- **Freshness.** An instance $\Psi_{SD}^{l_1}$ or $\Psi_{GW}^{l_2}$ is fresh, if $\mathcal{A}$ cannot derive the established session key shared between two participating entities after applying the Reveal(Ψ^l) query as well as $Test(\Psi^l)$ query provided in Table 6.2.

Under the ROR model, $\mathcal{A}$ has access to all the queries tabulated in Table 6.2. Furthermore, a "collision-resistant one-way cryptographic hash function $h(\cdot)$" is accessible to all the participants (SD_i, GW_j, ES_l, CS_k) including $\mathcal{A}$ as suggested in [42], and subsequently, we can model $h(\cdot)$ as a random oracle, say $Hash$.

Table 6.2: Different queries and their purposes

Query	Purpose
$Execute(\Psi_{SD}^{l_1}, \Psi_{GW}^{l_2})$	$\mathcal{A}$ executes this query to eavesdrop the messages exchanged between SD_i and GW_j
$CorruptSmartDevice(\Psi_{SD}^{l_1})$	$\mathcal{A}$ executes this query to extract "the credentials stored in memory of SD by its physical capturing"
$Reveal(\Psi^l)$	$\mathcal{A}$ executes this query to reveal a session key (the session key SK_{SD_i,GW_j} $(= SK_{GW_j,SD_i})$ between SD_i and GW_j or the session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$ between GW_j and ES_l, or the session key SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ between ES_l and CS_k) shared between Ψ^l and its respective partner
$Test(\Psi^l)$	$\mathcal{A}$ executes this query to test the reality of the session key between Ψ^l and its partner

The semantic security of the proposed CBACS-EIoT is now provided in Definition 6.1, which is helpful in proving Theorem 6.4.

Definition 6.1 (Semantic security). *If $Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly})$ represents the advantage of an adversary $\mathcal{A}$ in breaking the semantic security of CBACS-EIoT in time t_{poly} to derive the session key SK_{SD_i,GW_j} $(= SK_{GW_j,SD_i})$ between SD_i and GW_j or the session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$ between GW_j and ES_l, or the session key SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ between ES_l and CS_k, then $Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) = |2Pr[c' = c] - 1|$, where c and c' present the "correct" and "guessed" bits, respectively, and $Pr[E]$ is the probability of a random event E.*

Theorem 6.4. *Suppose an adversary $\mathcal{A}$ tries to derive the session key SK_{SD_i,GW_j} $(= SK_{GW_j,SD_i})$ between SD_i and GW_j or the session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$ between GW_j and ES_l, or the session key SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ between ES_l and CS_k in polynomial time t_{poly} in the proposed CBACS-EIoT. If q_h, $|Hash|$, and $Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$ represent "the number of Hash queries", "the range space of a one-way collision-resistant hash function $h(\cdot)$", and "the advantage of breaking the Elliptic Curve Decisional Diffe-Hellman Problem (ECDDHP)", respectively, $Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$ is expressed by the following:*

$$Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}).$$

Proof. The proof of this theorem is followed in similar manner as done in [42, 59, 137, 138, 188, 219]. According to the CBACS-EIoT, three games have been introduced, say $Game_l^{\mathcal{A}}$ for the adversary $\mathcal{A}$, $l = 0, 1, 2$. If $Succ_{Game_l}^{\mathcal{A}}$ is "an event that $\mathcal{A}$ can guess the random bit c in the game $Game_l^{\mathcal{A}}$ correctly", $\mathcal{A}$'s advantage (success probability) is winning $Game_l^{\mathcal{A}}$ in CBACS-EIoT can be then expressed as $Adv_{\mathcal{A},Game_l}^{CBACS-EIoT} = Pr[Succ_{Game_l}^{\mathcal{A}}]$. Each of the games is now described below.

- **Game$_0^{\mathcal{A}}$**: In this game, $\mathcal{A}$ executes the "actual attack" against CBACS-EIoT under the ROR model. Here, $\mathcal{A}$ picks a random bit c before $Game_0^{\mathcal{A}}$ begins. The semantic security defined in Definition 6.1 gives the following:

$$Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) = |2Adv_{\mathcal{A},Game_0}^{CBACS-EIoT} - 1|. \tag{6.9}$$

- **Game$_1^{\mathcal{A}}$**: $\mathcal{A}$ applies *eavesdropping attack* in this game to perform the *Execute* query in order to intercept all the communicated messages during the access control and key management phases. Suppose $\mathcal{A}$ intercepts the transmitted messages $\{Msg_{SG_1}, Msg_{SG_2}, Msg_{SG_3}\}$ during the access control phase between SD_i and GW_j to derive the session key SK_{GW_j,SD_i} $(= SK_{SD_i,GW_j})$, the messages $\{Msg_{GE_1}, Msg_{GE_2}, Msg_{GE_3}\}$ during access control between GW_j and ES_l to derive the session key SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$, and also the messages $\{Msg_{EC_1}, Msg_{EC_2}\}$ to construct the session key SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ during key management between ES_l and CS_k. Now, $\mathcal{A}$ applies *Reveal* and *Test* queries to validate if the derived session keys are correct ones or they are just random numbers. Since the temporal and long term secrets $\{TC_{SD_i}, PID_{SD_i}, K_{SD_i}, TC_{GW_j}, PID_{GW_j}, k_{GW_j}\}$ are needed to derive the session key SK_{GW_j,SD_i} $(= SK_{SD_i,GW_j})$, However, these secrets are protected by $h(\cdot)$, and thus, only interception of the messages $\{Msg_{SG_1}, Msg_{SG_2}, Msg_{SG_3}\}$ will not at all contribute in increasing the success probability in deriving the session key. The similar situation happens for deriving the secret

keys SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$ and SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ by the adversary $\mathcal{A}$ due to long and short term secrets. The games $Game_0^{\mathcal{A}}$ and $Game_1^{\mathcal{A}}$ then turn out to be *indistinguishable* in the presence of an "eavesdropping attack". Hence, we arrive to the following condition:

$$Adv_{\mathcal{A},Game_1}^{CBACS-EIoT} = Adv_{\mathcal{A},Game_0}^{CBACS-EIoT}. \tag{6.10}$$

- $\mathbf{Game_2^{\mathcal{A}}}$: In this game, $\mathcal{A}$ plays an *active attack*, where $\mathcal{A}$ includes the simulations of $Hash$ and $CorruptSmartDevice$ queries, and also the ECDDHP. To derive the session key SK_{SD_i,GW_j}, SK_{GW_j,ES_l} and SK_{ES_l,CS_k}, $\mathcal{A}$ needs to calculate $SK_{SD_i,GW_j} = h(h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) \cdot R_{GW_j} \,||PID_{SD_i})$ $(= SK_{GW_j,SD_i})$, $SK_{GW_j,ES_l} = h(g_j(PID_{GW_j}, PID_{ES_l}) \,||DK_{jl})$ $(= SK_{ES_l,GW_j})$ and $SK_{ES_l,CS_k} = h(h(r_{ES_l} \,||PID_{ES_l} \,||PID_{CS_k} \,||TS_{ES_l} \,||k_{ES_l})|| \, h(r_{CS_k} \,||TS_{CS_k} \,||k_{CS_k}) \,||f(PID_{ES_l}, PID_{CS_k}))$ $(= SK_{CS_k,ES_l})$, respectively. Assume that $\mathcal{A}$ has all the intercepted messages $\{Msg_{SG_1}, Msg_{SG_2}, Msg_{SG_3}\}$, $\{Msg_{GE_1}, Msg_{GE_2}, Msg_{GE_3}\}$, and $\{Msg_{EC_1}, Msg_{EC_2}\}$. From the intercepted messages $\{Msg_{SG_1}, Msg_{SG_2}, Msg_{SG_3}\}$ to derive the session key SK_{SD_i,GW_j}, $\mathcal{A}$ needs to compute $R_{SD_i} = h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) \cdot G$ and $R_{GW_j} = h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}) \cdot G$. Since R_{SD_i} and R_{GW_j} are protected using the $h(\cdot)$, to derive these parameters $\mathcal{A}$ needs to solve the ECDDHP in t_{poly} (polynomial) time, which has advantage probability $Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$ of the adversary $\mathcal{A}$. Similarly, for the other session keys $\mathcal{A}$ needs to solve the ECDDHP in t_{poly} time with same probability $Adv_{\mathcal{A}}^{ECDDHP}(t_{poly})$. Furthermore, all hash values are random in nature due to the usage of current timestamps and random nonces in all the communicated messages during communication. In addition, by applying $CorruptSmartDevice$ query, $\mathcal{A}$ will have the credentials that will be helpful to derive the session keys. However, the credentials in each IoT smart device are distinct and unique. Hence, both the games $\mathbf{Game_1^{\mathcal{A}}}$ and $\mathbf{Game_2^{\mathcal{A}}}$ are both indistinguishable if we exclude the simulations of $Hash$ and $CorruptSmartDevice$ queries, and also exclude the ECDDHP. By applying the birthday paradox to find the hash collision and the advantage of solving ECDDHP, we get the following result:

$$|Adv_{\mathcal{A},Game_1}^{CBACS-EIoT} - Adv_{\mathcal{A},Game_2}^{CBACS-EIoT}| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}). \tag{6.11}$$

Now, all the queries are executed by $\mathcal{A}$, and it is only remaining for $\mathcal{A}$ to guess correctly a bit for winning the game $Game_2^{\mathcal{A}}$. Therefore, it is obvious that

$$Adv_{\mathcal{A},Game_2}^{CBACS-EIoT} = \frac{1}{2}. \tag{6.12}$$

Eq. (6.9) simplifies to the following derivation:

$$\frac{1}{2}Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) = |Adv_{\mathcal{A},Game_0}^{CBACS-EIoT} - \frac{1}{2}|. \tag{6.13}$$

The following derivation comes by simplifying Eqs. (6.10), (6.11), (6.12) and (6.13), and applying the triangular inequality:

$$\begin{aligned} \frac{1}{2}Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) &= |Adv_{\mathcal{A},Game_0}^{CBACS-EIoT} - Adv_{\mathcal{A},Game_2}^{CBACS-EIoT}| \\ &= |Adv_{\mathcal{A},Game_1}^{CBACS-EIoT} - Adv_{\mathcal{A},Game_2}^{CBACS-EIoT}| \\ &\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}). \end{aligned} \tag{6.14}$$

Finally, the final result is arrived by multiplying both sides of Eq. (6.14) by "a factor of 2":

$$Adv_{\mathcal{A}}^{CBACS-EIoT}(t_{poly}) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_{poly}).$$

□

6.5.3 Informal security analysis

In this section, we informally (non-mathematically) show that the proposed CBACS-EIoT is robust against the following potential attacks. In addition, we also discuss other functionality features that are offered by the proposed CBACS-EIoT.

Proposition 6.1. *CBACS-EIoT is resilient against replay attack.*

Proof. The communication messages involved during access control and the key management phases among different entities, namely IoT smart devices, gateway, edge server and cloud server are $Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}$, $Msg_{SG_2} = \{TID^*_{SD_i}, R_{GW_j}, Sign_j, TS_{GW_j}\}$, $Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS^*_{SD_i}\}$, $Msg_{GE_1} = \{TID_{GW_j}, R_{GW_j}, Sign_j, TS_{GW_j}\}$, $Msg_{GE_2} = \{PID^*_{ES_l}, TID^*_{GW_j}, R_{ES_l}, Sign_l, TS_{ES_l}\}$, $Msg_{GE_3} = \{SKV_{GW_j,ES_l}, TS^*_{GW_j}\}$, $Msg_{EC_1} = \{TID_{ES_l}, A_{ES_l}, TS_{ES_l}\}$ and $Msg_{EC_2} = \{B_{CS_k}, C_{CS_k}, D_{CS_k}, SKV_{CS_k,ES_l}, TS_{CS_k}\}$, respectively, as discussed in Sections 6.4.3 and 6.4.4. During the aforesaid individual message formation, unique timestamps and random nonces are eventually are appended with each message. For example, the message Msg_{SG_1} consists of TID_{SD_i}, R_{SD_i}, $Sign_i$, TS_{SD_i}, where TS_{SD_i} signifies the current timestamp of the system (smart device), and R_{SD_i} is a random nonce selected by SD_i. Moreover, both random numbers and timestamps are verified by the intended receiver(s) and transmitter(s) ends for substantiating the freshness of each message

during the session key initiation task. Thus, only replaying the old timestamp-concatenated messages is easily detected either by the transmitter and the receiver sides. Therefore, the proposed CBACS-EIoT protects "replay attack". □

Proposition 6.2. *CBACS-EIoT is resilient against man-in-the-middle(MiTM) attack.*

Proof. Suppose an adversary $\mathcal{A}$ intercepts a smart device (SD_i)'s request message $Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}$ from an open channel (see Section 6.4.3), and tries to originate another valid request message, say Msg'_{SG_1} on the fly for its associated gateway node GW_j so that the GW_j will be unable to detect it as an modified message. Since the adversary $\mathcal{A}$ does not have the knowledge about the pre-loaded secret credentials, namely TC_{SD_i}, and the random secrets k_{SD_i} and r_{SD_i}, he/she cannot build the valid request message Msg'_{SG_1}. Similarly, $\mathcal{A}$ is also unable to construct a valid SD_i's response message msg'_{SG_2} in lieu of the actual intercepted message $Msg_{SG_2} = \{TID^*_{SD_i}, R_{GW_j}, Sign_j, TS_{GW_j}\}$. Because $\mathcal{A}$ does not have the knowledge about the pre-loaded secret credentials (TC_{GW_j} and k_{GW_j}), and the chosen random secret r_{GW_j} and the computed session key SK_{GW_j,SD_i}. Moreover, $\mathcal{A}$ can not change SD_i's "acknowledgement message" $Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS^*_{SD_i}\}$ due to the formation of the authenticated shared secret session key SK_{GW_j,SD_i} between SD_i and GW_j. In the similar fashion, it is computationally intractable for $\mathcal{A}$ to compute the modified messages Msg'_{GE_1}, Msg'_{GE_2}, Msg'_{GE_3}, Msg'_{EC_1} and Msg'_{EC_2} without having the knowledge about the corresponding parameters r_{GW_j}, TC_{GW_j}, k_{GW_j}, r_{ES_l}, SK_{GW_j,ES_l}, k_{ES_l}, r_{CS_k}, SK_{ES_l,CS_k}, k_{CS_k}, $g_j(PID_{ES_l}, PID_{GW_j})$ and $f(PID_{ES_l}, PID_{CS_k})$. Thus, CBACS-EIoT is resilient against MiTM attack. □

Proposition 6.3. *CBACS-EIoT is secure against IoT smart device, gateway node, edge server and cloud server impersonation attacks.*

Proof. Suppose an adversary $\mathcal{A}$ tries to act himself/herself as a legitimate smart device SD_i to gateway node GW_j, and he (or she) wants to construct an authorization message, say $Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}$. In order to perform this operation, at first $\mathcal{A}$ needs to choose a random number $r'_{SD_i} \in Z^*_q$, and current timestamp TS'_{SD_i}, and later needs to compute $R'_{SD_i} = h(r'_{SD_i} ||TC_{SD_i} ||PID_{SD_i} ||TS'_{SD_i}) \cdot G$ and $Sign'_i = h(r'_{SD_i} ||TC_{SD_i} ||PID_{SD_i} ||TS'_{SD_i}) + h(Pub_{SD_i} ||Pub_{GW_j} ||TS'_{SD_i} ||TID_{SD_i} ||PID_{SD_i}) * k_{SD_i} \pmod q$. To make the smart device impersonation attack viable, $\mathcal{A}$ needs to proof that $R_{SD_i} = R'_{SD_i}$, and $Sign_i = Sign'_i$. But, these require the implicit knowledge about the concerned parameters, $TC_{SD_i}, ID_{SD_i}, PID_{SD_i}, mk_{SD_i}, mk_{RA}, mk_{GW_j}, RTS_{SD_i}$ and k_{SD_i}. Without knowing these parameters, it is computationally intractable to proof $R_{SD_i} = R'_{SD_i}$ and $Sign_i = Sign'_i$. $\mathcal{A}$ is

then unable to claim that he/she should be a legitimate IoT smart device to the gateway node GW_j. Hence, CBACS-EIoT is resilient against smart devices impersonation attack.

Assume $\mathcal{A}$ tries to act himself (or herself) as a legitimate gateway node (GW_j) to an edge server ES_l, and he (or she) wishes to create a valid message, say $Msg_{GE_1} = \{TID_{GW_j}, R_{GW_j}, Sign_j, TS_{GW_j}\}$. To do so, $\mathcal{A}$ can initially pick a random number r'_{GW_j} and current timestamp TS'_{GW_j} to calculate $R'_{GW_j} = h(TID_{GW_j}|| TC_{GW_j}|| PID_{GW_j}|| r'_{GW_j}|| k_{GW_j}|| TS_{GW_j}) \cdot G$ and $Sign'_j = h(TID_{GW_j} ||TC_{GW_j} ||PID_{GW_j} ||r'_{GW_j} ||k_{GW_j} ||TS'_{GW_j}) + h(TID_{GW_j} ||Pub_{GW_j} ||PID_{GW_j} ||TS_{GW'_j}) * k_{GW_j} \pmod q$. However, to make the impersonation attack feasible, $\mathcal{A}$ needs to show that $R_{GW_j} = R'_{GW_j}$ and $Sign_j = Sign'_j$. However, it needs the implicit knowledge about the concerned parameters, $TC_{GW_j}, ID_{GW_j}, PID_{GW_j}, mk_{GW_j}, mk_{RA}, RTS_{GW_j}$ and k_{GW_j}, and thus, without knowing these parameters, it is computationally hard to proof $R_{GW_j} = R'_{GW_j}$, and $Sign_j = Sign'_j$. Therefore, $\mathcal{A}$ is unable to claim that he/she should be a legitimate gateway node to ES_l, and CBACS-EIoT protects gateway node (GW_j) impersonation attack.

Suppose $\mathcal{A}$ attempts to act himself (or herself) as a legitimate edge server (ES_l) to cloud server CS_k, and he (or she) wants to create a valid request message, say $Msg_{EC_1} = \{TID_{ES_l}, A_{ES_l}, TS_{ES_l}\}$. In order to perform this task, $\mathcal{A}$ can pick a random number r'_{ES_l} and current timestamp TS'_{ES_l}, and then try to calculate $A'_{ES_l} = h(r'_{ES_l}||PID_{ES_l}||PID_{CS_k}||TS'_{ES_l}||k_{ES_l}) \oplus h(f(PID_{ES_l}, PID_{CS_k})||TS'_{ES_l})$. However, to make the impersonation attack feasible, $\mathcal{A}$ requires to proof that $A_{ES_l} = A'_{ES_l}$. Without the implicit knowledge about the concerned parameters, $ID_{ES_l}, ID_{CS_k} PID_{CS_k}, PID_{ES_l}, f(PID_{ES_l}, PID_{CS_k}), mk_{CS_k}, mk_{ES_l}$ and k_{ES_l}, it is computationally hard to proof $A_{ES_l} = A'_{ES_l}$. Thus, $\mathcal{A}$ is unable to claim that he (or she) should be a legitimate edge server to CS_k. As a result, CBACS-EIoT protects edge server (ES_l) impersonation attack.

Finally, consider that $\mathcal{A}$ tries to behave himself (or herself) as a legitimate cloud server (CS_k) to the edge server (ES_l), and he (or she) wants to construct a valid message, say $Msg_{EC_2} = \{B_{CS_k}, C_{CS_k}, D_{CS_k}, SKV_{CS_k,ES_l}, TS_{CS_k}\}$. To achieve this task, $\mathcal{A}$ can initially select a random number r'_{CS_k} and current timestamp TS'_{CS_k}, and then compute $SKV'_{CS_k,ES_l} = h(SK_{CS_k,ES_l}||TS'_{CS_k}||TID^{new}_{ES_l})$, $C'_{CS_k} = TID^{new}_{ES_l} \oplus h(TID_{ES_l}||SK_{CS_k,ES_l}||TS'_{CS_k})$, $B'_{CS_k} = h(r'_{CS_k}||TS'_{CS_k}||k_{CS_k}) \oplus h(f(PID_{CS_k}, PID_{ES_l})||TS'_{CS_k})$, and $D'_{CS_k} = PID_{CS_k} \oplus h(PID_{ES_l}||TID_{ES_l}||TS'_{CS_k})$. However, to make the impersonation attack feasible, $\mathcal{A}$ needs to satisfy $SKV_{CS_k,ES_l} = SKV'_{CS_k,ES_l}$, $C_{CS_k} = C'_{CS_k}$, $B_{CS_k} = B'_{CS_k}$, and $D_{CS_k} = D'_{CS_k}$, respectively. But, to assure the these conditions, $\mathcal{A}$ requires the implicit knowledge about the concerned parameters, $SK_{CS_k,ES_l}, ID_{ES_l}, ID_{CS_k} PID_{CS_k}, PID_{ES_l}, f(PID_{CS_k}, PID_{ES_l}), mk_{CS_k}, mk_{ES_l}$, and

k_{CS_k}. Hence, without knowing these parameters, it is also a computationally infeasible task for $\mathcal{A}$ to compute the said four parameters. This means that $\mathcal{A}$ is unable to claim that he (or she) should be a legitimate cloud server to ES_l. Hence, CBACS-EIoT also protects cloud server (CS_k) impersonation attack. □

Proposition 6.4. *CBACS-EIoT is secure against privileged-insider attack.*

Proof. The enrollment phase of individual principal with the RA is achieved in offline mode as discussed in Sections 6.4.2, 6.4.2, 6.4.2 and 6.4.2. Also, any other entity is not permitted to send its registration information to the trusted RA. In spite of this, the trusted RA alone originates all the credentials for individual principal in offline prior to their deployment in the IoT environment. Later, the RA encapsulates the corresponding information and stores them into each principal as specified in Section 6.4.2. Such a setting prevents a privileged-insider user of the RA, acting as an adversary, to avail any credentials that are pre-deployed in the memory of the individual entity. Hence, there does not exist any possibility of the privileged-insider attack in our proposed CBACS-EIoT. □

Proposition 6.5. *CBACS-EIoT is secure against Ephemeral Secret Leakage (ESL) attack.*

Proof. During the access control phase between a smart device and gateway node, the smart device SD_i computes the session key SK_{SD_i,GW_j} shared with its associated gateway node GW_j as $SK_{SD_i,GW_j} = h(h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) \cdot R_{GW_j} \,||PID_{SD_i})$, where $TC_{SD_i} = h(ID_{SD_i} \,||mk_{SD_i} \,||mk_{RA} \,||mk_{GW_j} \,||RTS_{SD_i})$ and $R_{GW_j} = h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}) \cdot G$. On the other hand, GW_j also calculates the session key SK_{GW_j,SD_i} shared with SD_i as $SK_{GW_j,SD_i} = h(h(r_{GW_j} \,||TC_{GW_j} \,||TID_{SD_i} \,||PID_{SD_i} \,||PID_{GW_j} \,||TS_{GW_j}) \cdot R_{SD_i} \,||PID_{SD_i})$, where $TC_{GW_j} = h(ID_{GW_j} \,||mk_{GW_j} \,||RTS_{GW_j} \,||mk_{RA})$ and $R_{SD_i} = h(r_{SD_i} \,||TC_{SD_i} \,||PID_{SD_i} \,||TS_{SD_i}) \cdot G$. From Theorem 6.1, it follows that SK_{SD_i,GW_j} $(= SK_{GW_j,SD_i})$. Similarly, from Theorems 6.2 and 6.3, the distinct session keys SK_{GW_j,ES_l} $(= SK_{ES_l,GW_j})$ and SK_{ES_l,CS_k} $(= SK_{CS_k,ES_l})$ are shared between a gateway node GW_j and an edge server ES_l, and between ES_l and a cloud server CS_k, respectively.

Intuitively, it can be easily seen from Section 6.5.2 that the "session key" is the arrangement of both session-centric (ephemeral) credentials (also called as "short-term secrets"), which are mainly the random numbers and timestamps, and the "long-term secrets" (temporal credentials, master keys, pseudo identities, and bivariate polynomial shares). The session keys SK_{SD_i,GW_j}, SK_{GW_j,ES_l} and SK_{ES_l,CS_k} can only be derived when an adversary $\mathcal{A}$ would compromise both the session-centric as well as long-term secrets. Moreover, utilization of

both random numbers and timestamps in session keys computation between SD_i and GW_j or between GW_j and ES_l or between ES_l and CS_k over different sessions forms always distinct session keys. In addition, for a particular session, even if any of the session keys is disclosed to $\mathcal{A}$, it will not effect in computing the session keys over other sessions. It happens due to the utilization of short-term and long term secrets during session keys formation. Hence, CBACS-EIoT is resilient against "session-temporary information attack". Meanwhile, CBACS-EIoT also preserves the "perfect forward and backward secrecy" property. Combining these, we conclude that CBACS-EIoT is secure against "ESL attack". □

Proposition 6.6. *CBACS-EIoT provides mutual authentication among various entities involved in the IoT network.*

Proof. During the access control phase between an IoT smart device (SD_i) and its respective gateway node (GW_j) as presented in Section 6.4.3, SD_i checks its GW_j's legitimacy before construction of a session secret key and vice versa. In such a provision, GW_j initially verifies the freshness of SD_i's request message (Msg_{SG_1}) by substantiating the timestamp received in the same message. Later, GW_j checks the signature ($Sign_i$: computed on r_{SD_i} utilizing the private key (k_{SD_i}) of the smart IoT device) of SD_i by verifying the condition: $Sign_i \cdot G \stackrel{?}{=} R_{SD_i} + h(Pub_{SD_i} || Pub_{GW_j} || TS_{SD_i} || TID_{SD_i} || PID_{SD_i}) \cdot Pub_{SD_i}$. If both the conditions are substantiated successfully, GW_j establishes a secret shared session key with SD_i. Similarly, SD_i also forms the shared secret session key with GW_j. Furthermore, to justify that the two intended parties (GW_j and SD_i) exchange the same shared secret session key ($SK_{GW_j,SD_i} = SK_{SD_i,GW_j}$) or not, a session key verifier (SKV_{SD_i,GW_j}) computation (at device end) and validation (at GW_j-side) approach is put forward. This ensures that both SD_i and GW_j share the same shared secret key for a particular session.

In the similar fashion, both GW_j and ES_l construct their shared secret session key $SK_{GW_j,ES_l} = SK_{ES_l,GW_j}$ followed by a successful mutual authentication task. More precisely, ES_l first checks the freshness of GW_j's request message (Msg_{GE_1}) by verifying the received timestamp appended into the same message. Later, ES_l verifies the signature ($Sign_j$: computed on r_{GW_j} by utilizing the private key (k_{GW_j}) of the gateway node) received from GW_j by checking the condition: $Sign_j \cdot G \stackrel{?}{=} R_{GW_j} + h(TID_{GW_j} || Pub_{GW_j} || PID_{GW_j} || TS_{GW_j}) \cdot Pub_{GW_j}$. If both the conditions are verified successfully, ES_l establishes a secret shared session key with GW_j. The substantiation of ES_l's legitimacy to GW_j is as similar as the verification process between SD_i and GW_j.

During the key management phase between an edge server (ES_l) and its associated cloud server (CS_k) as discussed in Section 6.4.3, CS_k checks the legitimacy of ES_l before

establishment of a session secret key and vice versa. CS_k first checks the freshness of ES_l's request message (Msg_{EC_1}) by utilizing the timestamps embedded in the same message. ES_l also checks the legitimacy of CS_k by verifying the timestamps from the received response message acknowledged by CS_k. Later, during session key formation phase, both CS_k and ES_l need to show their ability to decrypt the session-specific random challenges, such as $B_{CS_k}(= h(r_{CS_k}||TS_{CS_k}||k_{CS_k}) \oplus h(f(PID_{CS_k}, PID_{ES_l})||TS_{CS_k}))$ and $A_{ES_l}(= h(r_{ES_l}||PID_{ES_l}||PID_{CS_k}||TS_{ES_l}||k_{ES_l}) \oplus h(f(PID_{ES_l}, PID_{CS_k})||TS_{ES_l}))$ by utilizing their pre-shared polynomial shares. If both the entities are able to decrypt the random challenges, ES_l and CS_k establish a secret shared session key between them. Therefore, CBACS-EIoT provides a robust mutual authentication among different entities involved in the network. □

Proposition 6.7. *CBACS-EIoT provides strong block verification.*

Proof. In CBACS-EIoT, the block mining by the peer nodes (edge servers ES_l) in the P2P ES network is based on the widely-recognized "Ripple Protocol Consensus Algorithm (RPCA)". Assume a verifier, say $\mathcal{V}$ wants to verify a block present in the blockchain, which can be public, private or hybrid type. Without any loss of generality, suppose $\mathcal{V}$ wishes to verify a public-type block, say $Block_p = \{BV, PBHash, MTR, Public, TS, ES_l, Pub_{ES_l}, \{Tx_i \ |i = 1, 2, \cdots, n_t\}, CBHash, BSign\}$ as shown in Figure 6.11. $\mathcal{V}$ first requires to calculate the Merkle tree root (MTR') on the transactions stored in the block $Block_p$. If the recomputed MTR' does not match with the stored MTR in $Block_p$, $\mathcal{V}$ discards the $Block_m$. Otherwise, $\mathcal{V}$ needs to calculate block hash $CBHash'$ as $CBHash^* = h(BV||\ PBHash||\ MTR^*||\ Public||\ TS||\ ES_l||\ Pub_{ES_l}||\ Tx_1||\ Tx_2||\cdots||Tx_{n_t})$. If the calculated $CBHash'$ does not match with the stored $CBHash$, $\mathcal{V}$ also discards the $Block_m$. Finally, $\mathcal{V}$ verifies the block signature $BSign$ by applying using the ECDSA signature verification algorithm using the public key Pub_{ES_l} of the signer ES_l. If the signature validation takes place successfully, $\mathcal{V}$ accepts the block $Block_p$ as legitimate one. Thus, it is worth to observe that verification of a block in the blockchain is done based on a three-level verification process. □

Proposition 6.8. *CBACS-EIoT is resilient against IoT smart devices physical capture attack.*

Proof. In the presence of an unfriendly territory, there may be a chance to physically capture of few IoT smart devices (SD_i) by an adversary $\mathcal{A}$ as discussed in the proposed threat model (Section 6.3.2). Meanwhile, $\mathcal{A}$ can extract all the pre-loaded credentials $\{TID_{SD_i}, PID_{SD_i}, TC_{SD_i}, (k_{SD_i}, Pub_{SD_i}), E_q(u, v), G, h(\cdot)\}$ from a physically captured IoT smart device SD_i's memory by utilizing the "power analysis attacks" [141]. The secret credentials $\{PID_{SD_i},$

TC_{SD_i}, $(k_{SD_i}, Pub_{SD_i})\}$ stored in SD_i are unique from the credentials pre-deployed in other smart devices, SD_m. Again, the pseudo identity PID_{SD_i} and temporal credential TC_{SD_i} of an IoT smart device SD_i is computed by utilizing a "cryptographic one-way hash function" by the registration authority RA in offline mode. Therefore, it is not possible for $\mathcal{A}$ to reveal the master keys mk_{SD_i}, mk_{RA} and mk_{GW_j} and original identity of SD_i. As a result, physical compromise of the secret credentials pre-deployed in SD_i's memory can not lead the formation of the session keys among other non-captured smart IoT devices as well as the non-compromised IoT devices and their associated gateway nodes. In other words, the non-compromised smart devices can still able to communicate with their gateway nodes GW_j with 100% security. This scenario is known as "unconditionally secure" against node capture attack . Therefore, CBACS-EIoT is resilient against "IoT smart devices physical capture attack". □

Proposition 6.9. *CBACS-EIoT preserves both anonymity and untraceability properties.*

Proof. During the access control process between an IoT smart device (SD_i) and a gateway node (GW_j) in a particular application described in Section 6.4.3, SD_i sends the message $Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}$. GW_j then generates a new temporary identity $TID^{new}_{SD_i}$ for SD_i, computes $TID^{*}_{SD_i} = TID^{new}_{SD_i} \oplus h(SK_{GW_j,SD_i} ||TID_{SD_i} ||TS_{GW_j})$, and then sends the message $Msg_{SG_2} = \{TID^{*}_{SD_i}, R_{GW_j}, Sign_j, TS_{GW_j}\}$ to SD_i. Meanwhile, after receiving the message Msg_{SG_2} from GW_j, SD_i also computes $TID^{new}_{SD_i} = TID^{*}_{SD_i} \oplus h(SK_{SD_i,GW_j}$ $||TID_{SD_i} ||TS_{GW_j})$, updates TID_{SD_i} with the newly computed $TID^{new}_{SD_i}$ in its memory, and sends the message $Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS^{*}_{SD_i}\}$ to GW_j. GW_j also updates TID_{SD_i} with $TID^{new}_{SD_i}$ corresponding to PID_{SD_i} in its database in order to sync with SD_i. Since TID_{SD_i} is updated with $TID^{new}_{SD_i}$, an adversary even intercepts the messages Msg_{SG_1}, Msg_{SG_2} and Msg_{SG_3}, he/she can not linked TID_{SD_i} with $TID^{new}_{SD_i}$ with the messages exchanged between SD_i and GW_j in the next session. Moreover, all the components in the messages Msg_{SG_1}, Msg_{SG_2} and Msg_{SG_3} are dynamic and distinct, because they are associated with the random numbers (secrets) and current timestamps. The similar arguments work for the messages Msg_{GE_1}, Msg_{GE_2} and Msg_{GE_3} for the access control process between a gateway node (GW_j) and an edge server (ES_l) in a particular application described in Section 6.4.3, and the messages Msg_{EC_1}, Msg_{EC_2} and Msg_{EC_3} during the key management process between an edge server (ES_l) and a cloud server (CS_k) described in Section 6.4.4. Hence, we can say that the proposed CBACS-EIoT preserves both anonymity and untraceability properties at the same time. □

```
%%% Registration Authority (RA)
role registrationauthority(RA, SD, GW: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: RA
played_by RA
def=
local State: nat,
SKrasd, SKragw: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
G,MKra, MKsdi, Ksdi, Pubsdi, TCsdi, PIDsdi, RTSsdi, IDsdi, TIDsdi : text,
MKgwj, Kgwj, Pubgwj, TCgwj, PIDgwj, IDgwj, RTSgwj, TIDgwj  : text
const sp1, sp2: protocol_id
init  State := 0
transition
% smart device registration
1. State = 0  /\ Rcv(start) =|>
State' := 1   /\ Ksdi'   := new()
/\ Pubsdi' := F(Ksdi'.G)
/\ TCsdi'  := H(IDsdi.MKsdi.MKra.MKgwj.RTSsdi)
/\ PIDsdi' := H(IDsdi.MKsdi)
% Send registration message to RA
/\ Snd({TIDsdi.PIDsdi'.TCsdi'.Ksdi'}_SKrasd)
/\ secret({IDsdi,Ksdi',RTSsdi,MKsdi,MKgwj}, sp1, {RA})
%%Gatewaynode registration phase
/\ Kgwj'   := new()
/\ Pubgwj' := F(Kgwj'.G)
/\ TCgwj'  := H(IDgwj.MKgwj.RTSgwj.MKra)
/\ PIDgwj' := H(IDgwj.MKgwj)
% Send registration message to RA
/\ Snd({TIDgwj.PIDgwj'.TCgwj'.Kgwj'.TIDsdi.PIDsdi}_SKragw)
/\ secret({IDgwj,Kgwj',MKgwj,MKra}, sp2, {RA})
end role
```

Figure 6.14: HLSPL Specification for the role of the *RA* (Case 1)

6.6 Formal security verification via AVISPA tool: simulation study

In recent years, the formal security verification using automated tools becomes an important component in proving the security of a security protocol. The "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [17] is one of the widely-used automated software tools, which tests whether a security protocol is "safe" or "unsafe" or "inconclusive (may or may not be safe)" against passive/active attacks. Detailed discussions on AVISPA are provided in Chapter 2 (see Section 2.3.1).

```
%%% smart device SDi
role smartdevice(RA, SD, GW: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Smart Device SD
played_by SD
def=
local State: nat,
SKrasd: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
G, MKra, TIDsdi, IDsdi, MKsdi, Rsdi, Rsdi1, TCsdi, PIDsdi, TSsdi, RTSsdi,
Signi, Ksdi, TIDnew, TSsdi1  : text,
SKVsdigwj :text,
TIDgwj, IDgwj, RTSgwj, Kgwj, MKgwj, Rgwj, TSgwj  : text
const sp1, sp2, sdi_gwj_rsdi, sdi_gwj_tssdi, sdi_gwj_tssdi1, gwj_sdi_rgwj,
gwj_sdi_tsgwj : protocol_id
init  State := 0
transition
%%Smart device registration phase
%%% receive registration message securely from the RA
1. State = 0 /\ Rcv({TIDgwj.H(IDgwj.MKgwj).H(IDgwj.MKgwj.RTSgwj.MKra).Kgwj'.
TIDsdi.H(IDsdi.MKsdi)}_SKrasd) =|>
State' := 2  /\  secret({IDsdi,Ksdi',RTSsdi,MKsdi,MKgwj}, sp1, {RA})
%%%%% Access control phase
/\ Rsdi'  := new() /\ TSsdi' := new()
/\ Rsdi1'  := F(H(Rsdi'.TCsdi.PIDsdi.TSsdi').G)
/\ Signi' := Add(H(Rsdi'.H(IDsdi.MKsdi.MKra.MKgwj.RTSsdi).H(IDsdi.MKsdi).TSsdi').
H(F(Ksdi'.G).F(Kgwj'.G).TSsdi'.TIDsdi.H(IDsdi.MKsdi)).Kgwj')
%%% Send message Msg1 to GWj via public channel
/\ Snd(TIDsdi.Rsdi1'.Signi'.TSsdi)
%% SDi has freshly generated the values rdr and TSdr for GW that are included in Msg1
/\ witness(SD, GW, sdi_gwj_rsdi, Rsdi')
/\ witness(SD, GW, sdi_gwj_tssdi, TSsdi')

%%% Receive message Msg2 from the GW via public channel
2. State = 2 /\ Rcv(xor(TIDnew.H(H(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).
TIDsdi.H(IDsdi.MKsdi).
H(IDgwj.MKgwj).TSgwj').F(H(Rsdi'.TCsdi.H(IDsdi.MKsdi).TSsdi').G).
H(IDsdi.MKsdi)).TIDsdi.TSgwj')).F(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).
TIDsdi.H(IDsdi.MKsdi).H(IDgwj.MKgwj).TSgwj').G). Add(H(Rgwj'.H(IDgwj.
MKgwj.RTSgwj.MKra).TIDsdi.H(IDsdi.MKsdi). H(IDgwj.MKgwj).TSgwj').
H(TIDsdi.H(IDsdi.MKsdi).F(Ksdi'.G).F(Kgwj'.G).H(H(Rgwj'.H(IDgwj.
MKgwj.RTSgwj.MKra).TIDsdi.H(IDsdi.MKsdi).H(IDgwj.MKgwj).TSgwj').
F(H(Rsdi'.TCsdi.H(IDsdi.MKsdi).TSsdi').G).H(IDsdi.MKsdi)).
TSgwj').Kgwj').TSgwj') =|>
State' := 4 /\ TSsdi1' := new()
/\ SKVsdigwj' := H(H(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).TIDsdi.H(IDsdi.MKsdi).
H(IDgwj.MKgwj).TSgwj').F(H(Rsdi'.TCsdi.H(IDsdi.MKsdi).TSsdi').G).
H(IDsdi.MKsdi)).TSsdi1')
%%% Send message Msg3 to GWj via public channel
/\ Snd(SKVsdigwj'.TSsdi1')
%% SDi has freshly generated the value TSsdi1 for GWj that is included in Msg3
/\ witness(SD, GW, sdi_gwj_tssdi1, TSsdi1')
% SDi acceptance of the values rsdi, tssdi for GW by SD
/\ request(GW, SD, gwj_sdi_rgwj, Rgwj')
/\ request(GW, SD, gwj_sdi_tsgwj, TSgwj')
end role
```

Figure 6.15: HLSPL Specification for the role of smart device SD_i (Case 1)

```
role gatewaynode(RA, SD, GW: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Smart Device GW
played_by GW
def=
local State: nat,
SKragw: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
G,TIDgwj, IDgwj, MKgwj, MKra, Kgwj, RTSgwj, TSgwj, Rgwj,
RG, RGwj1, SKgwsdi: text,
TIDnew, TIDsdi, IDsdi, MKsdi, TCsdi, TSsdi, Rsdi, RTSsdi, Ksdi,
TIDsdi1, Signj, TSsdi1 : text
const sp1, sp2, sdi_gwj_rsdi, sdi_gwj_tssdi, sdi_gwj_tssdi1, gwj_sdi_rgwj,
gwj_sdi_tsgwj : protocol_id
init  State := 0
transition
%% GSS registration phase
%%% receive registration message securely from the CR
1. State = 0 /\ Rcv({TIDgwj.H(IDgwj.MKgwj).H(IDgwj.MKgwj.RTSgwj.MKra).Kgwj'.
TIDsdi.H(IDsdi.MKsdi)}_SKragw)  =|>
State' := 3 /\ secret({IDgwj,Kgwj',MKgwj,MKra}, sp2, {RA})
%% Access control Phase
%%% Received message Msg1 from the SD via public channel
2. State = 3 /\ Rcv(TIDsdi.F(H(Rsdi'.TCsdi.H(IDsdi.MKsdi).TSsdi').G).
Add(H(Rsdi'.H(IDsdi.MKsdi.MKra.MKgwj.RTSsdi).H(IDsdi.MKsdi).TSsdi').
H(F(Ksdi'.G).F(Kgwj'.G).TSsdi'.TIDsdi.H(IDsdi.MKsdi)).Kgwj'),TSsdi)=|>
%%TIDsdi,Rsdi',Signi',TSsdi  from Msg1
State' := 5 /\ Rgwj' := new() /\ TSgwj' := new()
/\ RGwj1' := F(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).TIDsdi.
H(IDsdi.MKsdi).H(IDgwj.MKgwj).TSgwj').G)
/\ SKgwsdi' := H(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).TIDsdi.H(IDsdi.MKsdi).
H(IDgwj.MKgwj).TSgwj').F(H(Rsdi'.TCsdi.H(IDsdi.MKsdi).TSsdi').G).
H(IDsdi.MKsdi))
/\ TIDsdi1' := xor(TIDnew.H(SKgwsdi'.TIDsdi.TSgwj'))
/\ Signj'  := Add(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).TIDsdi.H(IDsdi.MKsdi).
H(IDgwj.MKgwj).TSgwj'). H(TIDsdi.H(IDsdi.MKsdi).F(Ksdi'.G).
F(Kgwj'.G).SKgwsdi'.TSgwj').Kgwj')
%%% Send message Msg2 to the Smart Device via public channel
/\ Snd(TIDsdi1'.RGwj1'.Signj'.TSgwj')
%% Gateway Node(GWj) has freshly generated Rgwj and TSgwj for smart device in Msg2
/\ witness(GW, SD, gwj_sdi_rgwj, Rgwj')
/\ witness(GW, SD, gwj_sdi_tsgwj, TSgwj')
%%% Receive message Msg3 from the smart device via public channel
3. State = 5 /\ Rcv(H(H(H(Rgwj'.H(IDgwj.MKgwj.RTSgwj.MKra).T
IDsdi.H(IDsdi.MKsdi).H(IDgwj.MKgwj).TSgwj'). F(H(Rsdi'.TCsdi.
H(IDsdi.MKsdi).TSsdi').G).H(IDsdi.MKsdi)).TSsdi1') .TSsdi1') =|>
% GWj acceptance of the values (message Msg3) for GW by SDi
State' := 7 /\ request(SD, GW, sdi_gwj_rsdi, Rsdi')
/\ request(SD, GW, sdi_gwj_tssdi, TSsdi')
/\ request(SD, GW, sdi_gwj_tssdi1, TSsdi1')
end role
```

Figure 6.16: HLSPL Specification for the role of gateway node GW_j (Case 1)

```
%%% Role for the session
role session (RA, SD, GW: agent, H: hash_func)
def=
local  Sn1, Sn2, Sn3, Rv1, Rv2, Rv3: channel (dy)
composition
registrationauthority (RA, SD, GW, H, Sn1, Rv1)
/\ smartdevice (RA, SD, GW, H, Sn2, Rv2)
/\ gatewaynode (RA, SD, GW, H, Sn3, Rv3)
end role


%%% Role for the goal and environment
role environment()
def=
const ra, sd, gw: agent,
h, f, add, poly: hash_func,
tssdi, tssdi1, tsgwj: text,
sp1, sp2, sdi_gwj_rsdi, sdi_gwj_tssdi, sdi_gwj_tssdi1,
gwj_sdi_rgwj, gwj_sdi_tsgwj: protocol_id
intruder_knowledge = {ra, sd, gw, h, f, add, poly,
tssdi, tssdi1, tsgwj}
composition
session(ra, sd, gw, h)
/\ session(ra, i, gw, h)
/\ session(ra, sd, i, h)
end role

goal
%%% Confidentiality (privacy)
secrecy_of sp1, sp2
%%% Authentication
authentication_on sdi_gwj_rsdi,sdi_gwj_tssdi, sdi_gwj_tssdi1
authentication_on gwj_sdi_rgwj, gwj_sdi_tsgwj
end goal
environment()
```

Figure 6.17: HLSPL Specification for the role of the session, goal and environment (Case 1)

The designed security protocols need to tested AVISPA tool require to be specified in a language, called "HLPSL (High Level Protocols Specification Language)", which is a role-oriented language [17]. The implementation of a security protocol in HLPSL is coded in a file with extension "hlpsl". HLPSL contains two types of roles: a) the "basic roles" that are optional roles and used to represent each participant role, and b) the "composition roles" that are mandatory roles and mainly used to represent scenarios of the developed basic roles. It is worth noticing that each role is treated as an independent role from other roles, which

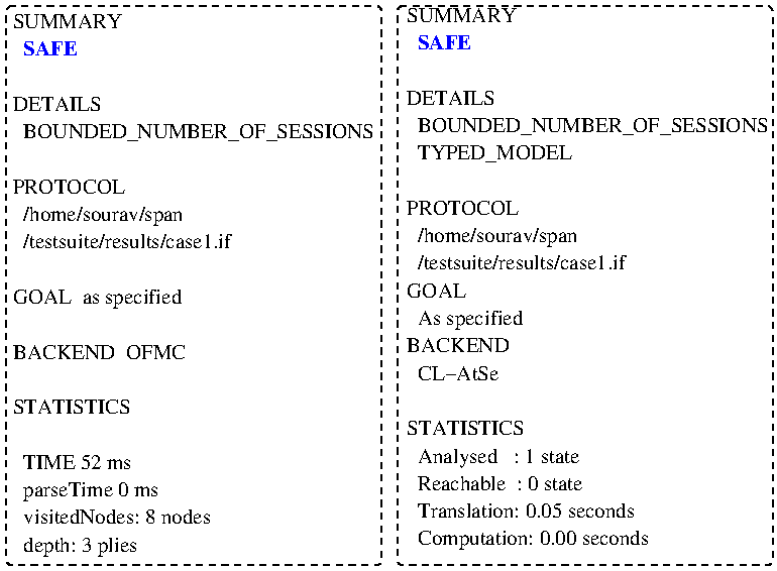

Figure 6.18: Simulation results of CBACS-EIoT (Case 1) under OFMC and CL-AtSe backends

receive some starting information by the parameters and then contact with other roles by the channels. In AVISPA, the channels are considered as the "Dolve-Yao (DY) threat model" [67] (as it is also defined in our threat model in Section 6.3.2) channels and therefore, an adversary can intercept, modify, update, delete of inject fake messages during communication over public channels. More detailed information on AVISPA and HLPSL can be found in [17].

In our implementation, we consider the following three cases:

- **Case 1**: In this scenario, we have defined the basic roles for the participants: the RA (see in Figure 6.14) "IoT smart device (SD_i)" (see in Figure 6.15) and "gateway node (GW_j)" (see in Figure 6.16), and the mandatory composition roles (session and goal & environment) during the access control phase between SD_i and GW_j (see in Figure 6.17).

- **Case 2**: In this scenario, we have defined the basic roles for the participants: the RA (see in Figure 6.19), "gateway node (GW_j)" (see in Figure 6.20) and "edge server (ES_l)" (see in Figure 6.21), and the mandatory composition roles (session and goal & environment) during the access control phase between GW_j and ES_l (see in Figure 6.22).

- **Case 3**: In this scenario, we have defined the basic roles for the participants: the RA (see Figure 6.24), "edge server (ES_l)" (see in Figure 6.25) and "cloud server (CS_k)" (see in Figure 6.26), and also the mandatory composition roles (session and goal &

```
%%% Role for the RA
role registrationauthority(RA, GW, ES: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: RA
played_by RA
def=
local State: nat,
SKragw, SKraes: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add: hash_func,
%%% Poly is polynomial function
Poly: hash_func,
Kgwj, IDgwj, Pubgwj, TCgwj, RTSgwj, PIDgwj : text,
MKgwj, MKra, TIDgwj, G : text,
IDesl, RTSesl, MKesl, TIDesl, TCesl, Pubesl,
PIDesl, Kesl : text
const sp1, sp2: protocol_id

init  State := 0
transition
%%Gateway node registration phase
1. State = 0 /\ Rcv(start) =|>
State' := 1 /\ Kgwj' := new() /\ Pubgwj' := F(Kgwj'.G)
/\ RTSgwj' := new()
/\ TCgwj' := H(IDgwj.MKgwj.RTSgwj'.MKra)
/\ PIDgwj' := H(IDgwj.MKgwj)
%%% Send registration message to GWj via secure channel
/\ Snd({TIDgwj.PIDgwj'.TCgwj'.Kgwj'}_SKragw)
/\ secret({IDgwj,RTSgwj',MKgwj,MKra}, sp1, {RA})
%%Edge Server registration phase
/\ Kesl' := new() /\ Pubesl' := F(Kesl'.G)
/\ RTSesl' := new()
/\ TCesl' := H(IDesl.MKesl.RTSesl'.MKra)
/\ PIDesl' := H(IDesl.MKesl)
%%% Send registration message to ESl via secure channel
/\ Snd({TIDesl.PIDesl'.TCesl'.Kesl'.TIDgwj.PIDgwj'}_SKraes)
/\ secret({IDesl,RTSesl',MKesl,MKra}, sp2, {RA})
end role
```

Figure 6.19: HLSPL Specification for the role of the RA (Case 2)

environment) during the key management phase between ES_l and CS_k (see in Figure 6.27).

We have then simulated the three cases defined above under the "Security Protocol ANimator for AVISPA (SPAN)" tool [18]. The simulation results for the three cases are illustrated in Figures 6.18, 6.23 and 6.28, respectively. It is worth noticing that we have applied only OFMC and CL-AtSe backends for the formal security verification in all the three cases. We have ex-

```
%%% Gateway node GW
role gatewaynode(RA, GW, ES: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: GW
played_by GW
def=
local State: nat, SKragw: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
Kgwj, IDgwj, Pubgwj, TCgwj, RTSgwj, PIDgwj, TSesl,
TIDgwj1, Resl, TSgwj1 : text,
MKgwj, MKra, TIDgwj, G : text,
IDesl, RTSesl, MKesl, TIDesl, TCesl, Pubesl,
PIDesl, Kesl : text,
SKVgwes, TSgwj, Rgwj, R1gwj, Sigj : text
const sp1, sp2, gw_es_rgw, gw_es_tsgw, gw_es_tsgw1,
es_gw_res, es_gw_tses : protocol_id
init  State := 0
transition
%%GW registration phase
%%% GW receive registration message securely from the RA
1. State = 0 /\ Rcv({TIDgwj.H(IDgwj.MKgwj).H(IDgwj.MKgwj.RTSgwj'.
MKra).Kgwj'}_SKragw) =|>
State' := 2 /\ secret({IDgwj,RTSgwj',MKgwj,MKra}, sp1, {RA})
%%% Access control phase
/\ Rgwj' := new() /\ TSgwj' := new()
/\ R1gwj' := F(H(TIDgwj.H(IDgwj.MKgwj.RTSgwj'.MKra).
H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G)
/\ Sigj' := Add(H(IDgwj.H(IDgwj.MKgwj.RTSgwj'.MKra).
H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').
H(TIDgwj.F(Kgwj'.G).H(IDgwj.MKgwj).TSgwj').Kgwj')
%%% GW send message Msg_GE1 to ES via public channel
/\ Snd(TIDgwj.R1gwj'.Sigj'.TSgwj')
%% GW has freshly generated rgwj and TSgwj for ES that are included in Msg1
/\ witness(GW, ES, gw_es_rgw, Rgwj')
/\ witness(GW, ES, gw_es_tsgw, TSgwj')
%%% ~~~~~~~~GW receive message Msg_GE2 from the ES via public channel
%%%~~~~~~~~Rcv(PIDesl'.TIDgwj11'.R1esl'.Sigl'.TSesl')
2. State = 2
/\ Rcv(xor(H(IDesl.MKesl).H(H(IDgwj.MKgwj).F(Kgwj'.G).F(Kesl'.G).TSesl')).
xor(TIDgwj1.H(TIDgwj.H(Poly(H(IDesl.MKesl).H(IDgwj.MKgwj)).F(H(H(IDgwj.MKgwj).
H(IDesl.MKesl.RTSesl'.MKra).H(IDesl.MKesl).Resl'.Kesl'.TSesl').
F(H(TIDgwj.H(IDgwj.MKgwj.RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G))).TSesl')).
F(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).H(IDesl.MKesl).Resl'.Kesl'.TSesl').G).
Add(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).H(IDesl.MKesl).Resl'.Kesl'.TSesl').
H(H(Poly(H(IDesl.MKesl).H(IDgwj.MKgwj)).F(H(H(IDgwj.MKgwj).
H(IDesl.MKesl.RTSesl'.MKra).H(IDesl.MKesl).Resl'.Kesl'.TSesl').F(H(TIDgwj.
H(IDgwj.MKgwj.RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G))).
H(IDgwj.MKgwj).F(Kgwj'.G).F(Kesl'.G)).Kesl').TSesl') =|>
State' := 4 /\ TSgwj1' := new()
/\ SKVgwes' := H(H(Poly(H(IDesl.MKesl).H(IDgwj.MKgwj)).
F(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).H(IDesl.MKesl).Resl'.Kesl'.TSesl').
F(H(TIDgwj.H(IDgwj.MKgwj.RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G))).TSgwj1')
%%% GW send message Msg_GE3 to ES via public channel
/\ Snd(SKVgwes'.TSgwj1')
%% GW has freshly generated the value TSgwj1 for ES that is included in Msg_GE3
/\ witness(GW, ES, gw_es_tsgwj1, TSgwj1')
% GW's acceptance of the values resl, TSesl (message Msg_GE2) for ES by GW
/\ request(ES, GW, es_gw_resl, Resl')
/\ request(ES, GW, es_gw_tsesl, TSesl')
end role
```

Figure 6.20: HLSPL Specification for the role of gateway node GW_j (Case 2)

```
%%% Role for the ES
role edgeserver(RA, GW, ES: agent, H: hash_func,Snd, Rcv: channel(dy))
played_by ES
def=
local State: nat, SKraes: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add: hash_func, Poly: hash_func,
Kgwj, IDgwj, Pubgwj, TCgwj, RTSgwj, PIDgwj, R1esl, Resl, TSesl,
TIDgwj1, TIDgwj11, TSgwj1  : text, MKgwj, MKra, TIDgwj, G : text,
IDesl, RTSesl, MKesl, TIDesl, TCesl, Pubesl, PIDesl, Kesl : text,
SKVgwes, TSgwj, Rgwj, R1gwj, Sigj, Sigl, DKlj, SKesgw : text
const sp1, sp2, dr_gss_rdr, dr_gss_tsdr, dr_gss_tsdr1,  gss_dr_rgss1,
gss_dr_tsgss: protocol_id
init  State := 0
transition
%% ES registration phase
%%% ES receive registration message securely from the RA
1. State = 0 /\ Rcv({TIDesl.H(IDesl.MKesl).H(IDesl.MKesl.RTSesl'.MKra).
Kesl'.TIDgwj.H(IDgwj.MKgwj)}_SKraes)  =|>
State' := 3 /\ secret({IDesl,RTSesl',MKesl,MKra}, sp2, {RA})
%%% Access control phase
%%% ES receive message Msg_GE1 from the GW via public channel %%
%% Rcv(TIDgwj.R1gwj'.Sigj'.TSgwj')
2. State = 3 /\ Rcv(TIDgwj.F(H(TIDgwj.H(IDgwj.MKgwj.RTSgwj'.MKra).
H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G). Add(H(TIDgwj.H(IDgwj.
MKgwj.RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').
H(TIDgwj.F(Kgwj'.G).H(IDgwj.MKgwj).TSgwj').Kgwj').TSgwj') =|>
State' := 5 /\ Resl' := new() /\ TSesl' := new()
/\ R1esl' := F(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).
H(IDesl.MKesl).Resl'.Kesl'.TSesl').G)
/\ DKlj' := F(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).
H(IDesl.MKesl).Resl'.Kesl'.TSesl').F(H(TIDgwj.H(IDgwj.MKgwj.
RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.TSgwj').G))
/\ SKesgw' := H(Poly(H(IDesl.MKesl).H(IDgwj.MKgwj)).DKlj')
/\ PIDesl' := xor(H(IDesl.MKesl).H(H(IDgwj.MKgwj).F(Kgwj'.G).
F(Kesl'.G).TSesl'))
/\ Sigl' := Add(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).
H(IDesl.MKesl).Resl'.Kesl'.TSesl').
H(SKesgw'.H(IDgwj.MKgwj).F(Kgwj'.G).F(Kesl'.G)).Kesl')
/\ TIDgwj11' := xor(TIDgwj1.H(TIDgwj.SKesgw'.TSesl'))
%%% ES send message Msg_GE2 to the GW via public channel
/\ Snd(PIDesl'.TIDgwj11'.R1esl'.Sigl'.TSesl')
%% ES has freshly generated resl and TSesl for GW in Msg_GE2
/\ witness(ES, GW, es_gw_resl, Resl')
/\ witness(ES, GW, es_gw_tsesl, TSesl')
%%% ES receive message Msg_GE3 from the GW via public channel
3. State = 5 /\ Rcv(H(H(Poly(H(IDesl.MKesl).H(IDgwj.MKgwj)).
F(H(H(IDgwj.MKgwj).H(IDesl.MKesl.RTSesl'.MKra).
H(IDesl.MKesl).Resl'.Kesl'.TSesl').F(H(TIDgwj.H(IDgwj.
MKgwj.RTSgwj'.MKra).H(IDgwj.MKgwj).Rgwj'.Kgwj'.
TSgwj').G))).TSgwj1').TSgwj1') =|>
% ES's acceptance of rgwj, TSgwj and TSgwj1 by GW
State' := 7 /\ request(GW, ES, gw_es_rgw, Rgwj')
/\ request(GW, ES, gw_es_tsgw, TSgwj')
/\ request(GW, ES, gw_es_tsgwj1, TSgwj1')
end role
```

Figure 6.21: HLSPL Specification for the role of edge server ES_l (Case 2)

```
%%% Role for the session
role session (RA, GW, ES: agent,
H: hash_func)
def=
local  Sn1, Sn2, Sn3, Rv1, Rv2, Rv3: channel (dy)
composition
registrationauthority(RA, GW, ES, H, Sn1, Rv1)
/\ gatewaynode(RA, GW, ES, H, Sn2, Rv2)
/\ edgeserver(RA, GW, ES, H, Sn3, Rv3)
end role

%%% Role for the goal and environment
role environment()
def=
const ra, gw, es: agent,
h, f, add, poly: hash_func,
tsgwj, tsgwj1, tsesl: text,
sp1, sp2, gw_es_tsgwj1, gw_es_tsgw, gw_es_rgw,
es_gw_resl, es_gw_tsesl: protocol_id
intruder_knowledge = {ra, gw, es, h, f, add,
poly, tsgwj, tsgwj1, tsesl}

composition
session(ra, gw, es, h)
/\ session(ra, i, es, h)
/\ session(ra, gw, i, h)
end role

goal
%%% Confidentiality (privacy)
secrecy_of sp1, sp2
%%% Authentication
authentication_on gw_es_tsgwj1, gw_es_tsgw, gw_es_rgw
authentication_on es_gw_resl, es_gw_tsesl
end goal
environment()
```

Figure 6.22: HLSPL Specification for the role of the session, goal and environment (Case 2)

cluded the simulation results under two other backends, namely SATMC and TA4SP. This is because these backends do not currently support implementing the "bitwise exclusive-OR (XOR)" operation, and the simulation results under these backends will appear as "inconclusive". The simulation results show that the proposed scheme for all the cases (Case 1, Case 2 and Case 3) is secure against both "replay" and "man-in-the-middle" attacks.

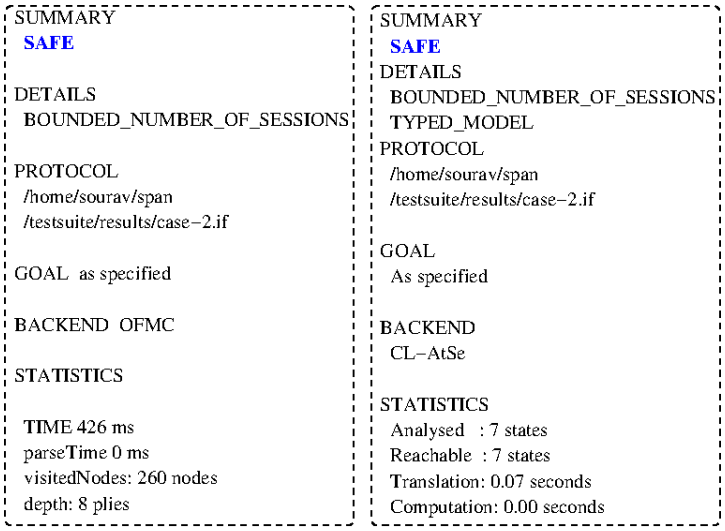

Figure 6.23: Simulation results of CBACS-EIoT (Case 2) under OFMC and CL-AtSe backends

Table 6.3: Communication costs comparative study

Protocol	Number of transmitted messages	Total communication cost (in bits)
Luo *et al.* [131]	2	3040
Li *et al.* [120]	2	3488
Aziz *et al.* [20]	5	4320
CBACS-EIoT (**Case CA_1**)	3	1728
CBACS-EIoT (**Case CA_2**)	3	1984
CBACS-EIoT (**Case CA_3**)	2	1504

6.7 Comparative analysis

This section provides a detailed comparative study on the following: a) "security and functionality features", b) communication costs, and c) computation costs, among the proposed CBACS-EIoT and other relevant existing schemes, such as the schemes suggested by Luo *et al.* [131], Li *et al.* [120], and Aziz *et al.* [20].

In this section, we take the following three cases for comparative analysis for communication and computation costs:

- **Case CA_1**: Access control phase between a smart device SD_i and its associated gateway node GW_j.

```
%%% Registration Authority (RA)
role registrationauthority(RA, ES, CS: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: RA
played_by RA
def=
local State: nat,
SKraes, SKracs: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add: hash_func,
G, Kesl, MKesl, Pubesl, TIDesl, TCesl, Kcsk, PIDcsk: text,
MKra, IDesl, TIDcsk: text,
RTSesl, PIDesl, Pubcsk, IDcsk, MKcsk: text
const sp1, sp2: protocol_id
init  State := 0
transition
%%Edge Server registration phase
1. State = 0 /\ Rcv(start) =|>
State' := 1 /\ Kesl'   := new() /\ RTSesl' := new()
/\ Pubesl' := F(Kesl'.G)
/\ TCesl'  := H(IDesl.MKesl.RTSesl'.MKra)
/\ PIDesl' := H(IDesl.MKesl)
%%% Send registration message ES to RA via secure channel
/\ Snd({TIDesl.PIDesl'.TCesl'.Kesl'}_SKraes)
/\ secret({IDesl,RTSesl',MKra,MKesl,Kesl'}, sp1, {RA})
%%% Cloud server registration server
/\ Kcsk' := new() /\ Pubcsk' := F(Kcsk'.G)
/\ PIDcsk' := H(IDcsk.MKcsk)
%%% Send registration message ES CS via secure channel
/\ Snd({Kcsk'.TIDesl.PIDesl'.TIDcsk.PIDcsk'}_SKraes)
/\ secret({IDcsk,MKra,MKcsk,Kcsk'}, sp2, {RA})
end role
```

Figure 6.24: HLSPL Specification for the role of the RA (Case 3)

- **Case CA_2**: Access control phase between a gateway node GW_j and its associated edge server ES_l.

- **Case CA_3**: Key management phase between an edge server ES_l and its associated cloud server CS_k.

6.7.1 Comparative study on communication costs

For comparative analysis on communication costs, we consider an identity, a random nonce (number), timestamp, an elliptic curve point and a hash value (we use SHA-256 [140] is used

```
%%% Edge server ESl
role edgeserver(RA, ES, CS: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Edge server ESl
played_by ES
def=
local State: nat,
SKraes: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
G, TIDesl, IDesl, MKesl, RTSesl, MKra, Kesl, Resl, TSesl: text,
Aesl, IDcsk, MKcsk, Rcsk, Kcsk, TIDnew, TIDcsk, SKcses,
TScsk, PIDcsk: text
const sp1, sp2, esl_csk_resl, esl_csk_tsesl, csk_esl_rcsk,
csk_esl_tscsk : protocol_id
init  State := 0
transition
%%Edge server registration phase
%%% receive registration message securely from the RA
1. State = 0 /\ Rcv({TIDesl.H(IDesl.MKesl).H(IDesl.MKesl.
RTSesl'.MKra).Kesl'}_SKraes) =|>
% Rcv with the parameter({TIDesl.PIDesl.TCesl'.Kesl'}_SKraes)
State' := 2  /\ secret({IDesl,RTSesl',MKra',MKesl',Kesl'}, sp1, {RA})
%%% Access control phase of edge server
/\ Resl' := new() /\ TSesl' := new()
/\ Aesl' := xor(H(Resl'.H(IDesl.MKesl).H(IDcsk.MKcsk).TSesl'.Kesl').
H(Poly(H(IDesl.MKesl).H(IDcsk.MKcsk)).TSesl'))
%%% Send message Msg1 to CS via public channel
/\ Snd(TIDesl.Aesl'.TSesl')
%% ES has freshly generated Resl and TSesl for CS in Msg1
/\ witness(ES, CS, esl_csk_resl, Resl')
/\ witness(ES, CS, esl_csk_tsesl, TSesl')
2. State = 2 /\ Rcv(xor(H(Rcsk'.TScsk'.Kcsk').H(Poly(H(IDcsk.MKcsk).
H(IDesl.MKesl)).TScsk')).xor(TIDnew.H(TIDesl.SKcses'.TScsk')).
xor(PIDcsk.H(H(IDesl.MKesl).TIDesl.TScsk')).H(H(Rcsk'.
H(IDesl.MKesl).H(IDcsk.MKcsk).TSesl'.Kesl').H(Rcsk'.TScsk'.
Kcsk'.Poly(H(IDcsk.MKcsk).H(IDesl.MKesl)))).TScsk') =|>
%%% ESl's acceptance on the values r_cks and TS_csk
State'  :=  4 /\ request(CS, ES,  csk_esl_rcsk, Rcsk')
/\ request(CS, ES, csk_esl_tscsk, TScsk')
end role
```

Figure 6.25: HLSPL Specification for the role of edge server ES_l (Case 3)

```
role cloudserver(RA, ES, CS: agent,
H: hash_func, Snd, Rcv: channel(dy))
% Player: Edge server CSk
played_by CS
def=
local State: nat,
SKraes: symmetric_key,
%%% F is ECC point mutiplication operation
F, Add, Poly: hash_func,
G,TIDnew  : text,
Kcsk, TIDesl, MKesl, IDesl, MKcsk, MKra,
Resl, IDcsk, TIDcsk: text,
RTSesl, TSesl, Kesl, Rcsk, Bcsk, TScsk,
SKcses: text,
SKVcses, Ccsk, Dcsk, PIDcsk: text
const sp1, sp2, esl_csk_resl, esl_csk_tsesl, csk_esl_rcsk,
csk_esl_tscsk : protocol_id
init  State := 0
transition
%%Edge server registration phase
%%% receive registration message securely from the RA
1. State = 0 /\ Rcv({Kcsk'.TIDesl.H(IDesl.MKesl).TIDcsk.
H(IDcsk.MKcsk)}_SKraes) =|>
State' := 3 /\ secret({IDcsk,MKra,MKcsk,Kcsk'}, sp2, {RA})
/\ secret({IDesl,RTSesl',MKra',MKesl',Kesl'}, sp1, {RA})
%%% Access control phase
%%% Receive message Msg1 from the ES via public channel
2. State = 3 /\ Rcv(TIDesl.xor(H(Resl'.H(IDesl.MKesl).
H(IDcsk.MKcsk).TSesl'.Kesl').H(H(IDesl.MKesl).
H(IDcsk.MKcsk).TSesl').TSesl')) =|>
State' := 5 /\ Rcsk' := new() /\ TScsk' := new()
/\ Bcsk' := xor(H(Rcsk'.TScsk'.Kcsk').H(Poly(H(IDcsk.MKcsk).
H(IDesl.MKesl)).TScsk'))
/\ SKcses' := H(H(Rcsk'.H(IDesl.MKesl).H(IDcsk.MKcsk).TSesl'
.Kesl').H(Rcsk'.TScsk'.Kcsk'.Poly(H(IDcsk.MKcsk).H(IDesl.MKesl))))
/\ SKVcses' := H(SKcses'.TScsk.TIDnew)
/\ Ccsk' := xor(TIDnew.H(TIDesl.SKcses'.TScsk'))
/\ Dcsk' := xor(PIDcsk.H(H(IDesl.MKesl).TIDesl.TScsk'))
%%% Send message Msg2 to ES via public channel
/\ Snd(Bcsk'.Ccsk'.Dcsk'.SKVcses'.TScsk')
%% CS has freshly generated  Rcsk and TScsk for ES  in Msg1
/\ witness(CS, ES,  csk_esl_rcsk, Rcsk')
/\ witness(CS, ES, csk_esl_tscsk, TScsk')
%%% CS's acceptanceon values r_esl and TS_esl
/\ request(ES, CS, esl_csk_resl, Resl')
/\ request(ES, CS, esl_csk_tsesl, TSesl')
end role
```

Figure 6.26: HLSPL Specification for the role of cloud server CS_k (Case 3)

```
%%% Role for the session
role session (RA, ES, CS: agent,
H: hash_func)
def=
local  Sn1, Sn2, Sn3, Rv1, Rv2, Rv3: channel (dy)
composition
registrationauthority (RA, ES, CS, H, Sn1, Rv1)
/\ edgeserver (RA, ES, CS, H, Sn2, Rv2)
/\ cloudserver (RA, ES, CS, H, Sn3, Rv3)
end role

%%% Role for the goal and environment
role environment()
def=
const ra, es, cs: agent,
h, f, add, poly: hash_func,
tsesl, tscsk: text,
sp1, sp2, esl_csk_resl, esl_csk_tsesl, csk_esl_rcsk,
csk_esl_tscsk: protocol_id
intruder_knowledge = {ra, es, cs, h, f, add, poly,
tsesl, tscsk}

composition
session(ra, es, cs, h)
/\ session(ra, i, cs, h)
/\ session(ra, es, i, h)
end role
goal
%%% Confidentiality (privacy)
secrecy_of sp1, sp2
%%% Authentication
authentication_on esl_csk_resl, esl_csk_tsesl
authentication_on csk_esl_rcsk, csk_esl_tscsk
end goal
environment()
```

Figure 6.27: HLSPL Specification for the role of the session, goal and environment (Case 3)

for blockchain technology), which need 160 bits, 160 bits, 32 bits, $(160 + 160) = 320$ bits and 256 bits, respectively. We assume that a message size in the schemes of Luo *et al.* [131] and Li *et al.* [120] as 1024 bits. We consider the 160-bit ECC because it provides the equivalent security level when it is compared with that for 1024-bit RSA public key cryptosystem [22].

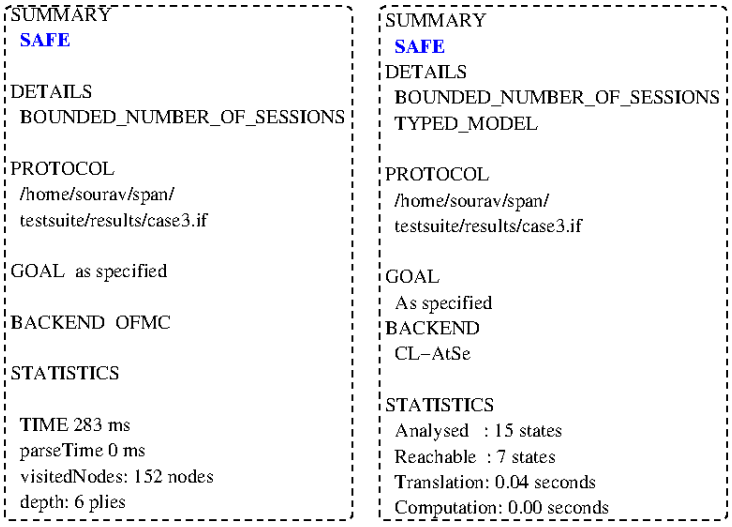

Figure 6.28: Simulation results of CBACS-EIoT (Case 3) under OFMC and CL-AtSe backends

Table 6.4: Execution costs (in milliseconds) required for different cryptographic operations [224]

Entity	T_{ecm}	T_{eca}	T_e	T_{bp}	T_h	T_m
IoT smart device (Mobile device)	13.405	0.081	2.249	32.713	0.056	0.008
Gateway/Server	2.165	0.013	0.339	5.427	0.007	0.001

The access control and key management phases are considered for the proposed CBACS-EIoT and other schemes [20, 120, 131]. In Case CA_1, three communicated messages $Msg_{SG_1} = \{TID_{SD_i}, R_{SD_i}, Sign_i, TS_{SD_i}\}$, $Msg_{SG_2} = \{TID^*_{SD_i}, R_{GW_j}, Sign_j, TS_{GW_j}\}$ and $Msg_{SG_3} = \{SKV_{SD_i,GW_j}, TS^*_{SD_i}\}$ require $(160+320+160+32) = 672$ bits, $(256+320+160+32) = 768$ bits, and $(256+32) = 288$ bits, respectively, which altogether require 1728 bits. In Case CA_2, three messages $Msg_{GE_1} = \{TID_{GW_j}, R_{GW_j}, Sign_j, TS_{GW_j}\}$, $Msg_{GE_2} = \{PID^*_{ES_l}, TID^*_{GW_j}, R_{ES_l}, Sign_l, TS_{ES_l}\}$ and $Msg_{GE_3} = \{SKV_{GW_j,ES_l}, TS^*_{GW_j}\}$ need $(160+320+160+32) = 672$ bits, $(256+256+320+160+32) = 1024$ bits and $(256+32) = 288$ bits, respectively, which altogether need 1984 bits. Furthermore, in Case CA_3, two messages $Msg_{EC_1} = \{TID_{ES_l}, A_{ES_l}, TS_{ES_l}\}$ and $Msg_{EC_2} = \{B_{CS_k}, C_{CS_k}, D_{CS_k}, SKV_{CS_k,ES_l}, TS_{CS_k}\}$ need $(160+256+32) = 448$ bits and $(256+256+256+256+32) = 1056$ bits, respectively, which altogether require 1504 bits. Table 6.3 illustrates comparison of communication costs among the proposed CBACS-EIoT and other schemes of Luo *et al.* [131], Li *et al.* [120] and Aziz *et al.* [20] in terms of number of transmitted messages and total communication costs in bits. It is evident that

CBACS-EIoT performs better in communication costs as compared to other schemes in all the cases.

6.7.2 Comparative study on computation costs

For comparative analysis on computation costs, the time needed for an "ECC point addition", an "ECC point multiplication", a "cryptographic one-way hash function", a "bilinear pairing operation", a "modular exponentiation operation", a "t-degree polynomial evaluation" and a "modular multiplication" are denoted by T_{ecm}, T_{eca}, T_h, T_{bp}, T_e, T_{poly} and T_m, respectively. The existing experimental results performed by Wu *et al.* [224] are used for rough computation costs. Wu *et al.* [224] used a "personal computer (Dell with an Intel Core i5-4460S@2.90GHz processor, 4GB main memory and the Window 8 operating system)" as the server, and a "personal mobile device (Samsung Galaxy S5 with a Quad-core 2.45G processor, 2GB memory and the Google Android 4.4.2 operating system)" as mobile device. Using the Horner's rule [110], the evaluation of a "t-degree uni-variate polynomial" needs t modular multiplications and t modular additions, and hence, $T_{poly} = tT_m + tT_a \approx tT_m$ by omitting modular addition operation as it is negligible as compared to "modular multiplication" operation. Approximate times needed for various cryptographic primitives based on experimental results reported in [224] are provided in Table 6.4.

The access control and key management phases are again considered for the proposed CBACS-EIoT and other schemes [20, 120, 131]. In Case CA_1, a smart device (SD_i) and a gateway node (GW_j) require $4T_{ecm} + T_{eca} + 5T_h \approx 53.981$ ms and $4T_{ecm} + T_{eca} + 5T_h \approx 8.708$ ms, respectively. In Case CA_2, GW_j and an edge server ES_l require $4T_{ecm} + T_{eca} + T_{poly} + 7T_h \approx (8.722 + 0.001t)$ ms and $4T_{ecm} + T_{eca} + T_{poly} + 7T_h \approx (8.722 + 0.001t)$ ms, respectively. In Case CA_3, ES_l and a cloud server CS_k need $T_{poly} + 7T_h \approx (0.049 + 0.001t)$ ms and $T_{poly} + 7T_h \approx (0.049 + 0.001t)$ ms, respectively. Even if the degree of bivariate polynomial is taken as $t = 1000$, in Case CA_2, total computation cost becomes 19.444 ms, and in Case CA_3, total computation cost becomes 2.098 ms. Table 6.5 tabulates the comparison of computation costs among the proposed CBACS-EIoT and other schemes [20, 120, 131]. It is observed that the proposed CBACS-EIoT requires less computation cost from the server side (gateway/edge server/cloud server) as compared to the schemes [120, 131]. Moreover, the computation cost of the proposed CBACS-EIoT at the smart device side is also comparable in that for other schemes [120, 131]. Though the computational costs needed for IoT smart device or user side and server side in the scheme of Aziz *et al.* [20] are lower as compared to those for our proposed CBACS-EIoT, it is justifiable because CBACS-EIoT provides significantly better

Table 6.5: Computation costs comparative study

Protocol	Smart device/user cost	Gateway/Server cost
Luo *et al.* [131]	$T_h + T_{bp}$ ≈ 32.769 ms	$3T_{ecm} + T_{eca} + 3T_{bp} + T_e + 3T_h$ ≈ 23.149 ms
Li *et al.* [120]	$T_h + T_{bp}$ ≈ 32.769 ms	$3T_{ecm} + 2T_{eca} + T_h + 4T_{bp}$ ≈ 28.236 ms
Aziz *et al.* [20]	$13T_h$ ≈ 0.728 ms	$4T_h$ ≈ 0.028 ms
CBACS-EIoT (**Case CA$_1$**)	$4T_{ecm} + T_{eca} + 5T_h$ ≈ 53.981 ms	$4T_{ecm} + T_{eca} + 5T_h$ ≈ 8.708 ms
CBACS-EIoT (**Case CA$_2$**)	–	$8T_{ecm} + 2T_{eca} + 2T_{poly} + 14T_h$ $\approx (17.444 + 0.002t)$ ms
CBACS-EIoT (**Case CA$_3$**)	–	$2T_{poly} + 14T_h$ $\approx (0.098 + 0.002t)$ ms

security and functionality features needed for a blockchain-based scheme, and also requires less communication costs as compared to the scheme of Aziz *et al.* [20].

6.7.3 Comparative study on functionality and security attributes

Finally, Table 6.6 shows a comparative analysis on various "functionality and security attributes" ($ATTR_1$–$ATTR_{15}$). From this table, it is evident that the proposed CBACS-EIoT supports all the attributes ($ATTR_1$–$ATTR_{15}$) listed there. On the other hand, both the schemes of Luo *et al.* [131] and Li *et al.* [120] do not support some attributes, such as $ATTR_2$, $ATTR_7$–$ATTR_{12}$, $ATTR_{14}$ and $ATTR_{15}$. In addition, Aziz *et al.*'s scheme [20] does not support the attributes $ATTR_8$–$ATTR_{10}$, $ATTR_{12}$ and $ATTR_{15}$. More importantly, the proposed CBACS-EIoT only supports the blockchain-based solution in order to keep transparency, immutability and decentralized properties.

Table 6.6: Comparison of functionality & security attributes

Attribute (ATTR)	Luo *et al.* [131]	Li *et al.* [120]	Aziz *et al.* [20]	CBACS-EIoT (all cases)
$ATTR_1$	✓	✓	✓	✓
$ATTR_2$	×	×	✓	✓
$ATTR_3$	✓	✓	✓	✓
$ATTR_4$	✓	✓	✓	✓
$ATTR_5$	✓	✓	✓	✓
$ATTR_6$	✓	✓	✓	✓
$ATTR_7$	×	×	✓	✓
$ATTR_8$	×	×	×	✓
$ATTR_9$	×	×	×	✓
$ATTR_{10}$	×	×	×	✓
$ATTR_{11}$	×	×	✓	✓
$ATTR_{12}$	×	×	×	✓
$ATTR_{13}$	✓	✓	✓	✓
$ATTR_{14}$	×	×	✓	✓
$ATTR_{15}$	×	×	×	✓

$ATTR_1$: "resilience against replay attack"; $ATTR_2$: support to mutual authentication; $ATTR_3$: support to key agreement; $ATTR_4$: "resilience against smart device impersonation attack"; $ATTR_5$: "resilience against gateway/server impersonation attack"; $ATTR_6$: "resilience against malicious device deployment attack"; $ATTR_7$: "resilience against smart device physical capture attack"; $ATTR_8$: "resilience against ESL attack under the CK-adversary model"; $ATTR_9$: support to anonymity property; $ATTR_{10}$: support to untraceability property; $ATTR_{11}$: "resilience against man-in-the-middle attack"; $ATTR_{12}$: "provide formal security analysis"; $ATTR_{13}$: "support to formal security verification using AVISPA tool"; $ATTR_{14}$: support to dynamic device addition phase"; $ATTR_{15}$: support to blockchain-based solution

✓: "a scheme is secure or it supports an attribute (ATTR)"; ×: "a scheme is insecure or it does not support an attribute (ATTR)".

6.8 Summary

In this article, we proposed a novel consortium blockchain-enabled access control scheme in edge computing based generic IoT environment (CBACS-EIoT). CBACS-EIoT not only supports access control among IoT smart devices and their associated gateway nodes, and among the gateway nodes and edge servers, but also key management process among the edge servers and the cloud servers. The transactions created from the data gathered security from the IoT smart devices by the gateway nodes are securely forwarded to the nearby edge nodes for

creating the blocks. The blocks are then mined by the nodes in the P2P ES network in order to verify and add them in the blockchain center. A detailed security analysis including the formal security under the random oracle model and formal security verification with the help of the broadly-used AVISPA automated software verification tool suggest that the proposed CBACS-EIoT can resist several potential attacks needed in a generic IoT environment. Moreover, a rigorous comparative study reveals that the proposed CBACS-EIoT provides better security and functionality attributes, and low communication cost and comparable computation cost as compared to those for other relevant schemes.

Chapter 7

Conclusion and Open Research Challenges

We first summarize the main contributions made in this thesis. Next, we also discuss some open research challenges that could be interesting for the security issues in IoT-related applicaions.

7.1 Contributions

In this thesis, we attemted to design three access control frameworks in order to provide the security in IoT-related environments for secure information sharing and data storage.

- We first designed a novel private blockchain-based access control protocol in IoT-enabled smart-grid system.
- Next, we suggested a new private blockchain envisioned access control system for securing industrial IoT (IIoT)-based pervasive edge computing.
- Finally, we proposed a novel consortium blockchain-enabled access control mechanism in edge computing based generic IoT environment.

The first proposal of the thesis supplied in **Chapter** 4 is to design a new blockchain-based access control protocol in IoT-enabled smart-grid system, called DBACP-IoTSG. Through the proposed DBACP-IoTSG, the data is securely brought to the service providers from their respectively smart meters. The P2P network is formed by the participating services providers, where the peer nodes are responsible for creating the blocks from the gathered data securely

from their corresponding smart meters and adding them into the blockchain after validation of the blocks using the voting-based consensus algorithm. In this work, the blockchain is considered as private because the data collected from the consumers of the smart meters are private and confidential. By the formal security analysis under the random oracle model, non-mathematical security analysis and software-based formal security verification, DBACP-IoTSG is shown to be resistant against various attacks. A detailed comparative study reveals that DBACP-IoTSG supports more functionality features and provides better security apart from its low communication and computation costs as compared to recently proposed relevant schemes. In addition, the blockchain implementation of DBACP-IoTSG has been performed to measure computational time needed for the varied number of blocks addition and also the varied number of transactions per block in the blockchain.

The second contribution of the thesis provided in **Chapter** 5 is to propose a new private blockchain-envisioned access control scheme for Pervasive Edge Computing (PEC) in IIoT environment, called PBACS-PECIIoT. We considered the private blockchain consisting of the transactions and registration credentials of the entities related to IIoT, because the information is strictly confidential and private. The security of PBACS-PECIIoT is significantly improved due to usage of blockchain. A meticulous comparative analysis exhibits that PBACS-PECIIoT achieves greater security and more functionality features, and requires low costs for communication and computational as compared to other pertinent schemes.

Finally, we have designed another novel consortium blockchain-enabled access control scheme in edge computing based generic IoT environment (called CBACS-EIoT) in **Chapter** 6. In CBACS-EIoT, the mutual authentication among the IoT smart devices and the gateway node(s), and also among the gateway node(s) and respective edge server(s) take place. In addition, key management phase is executed among the edge server(s) and associated cloud server(s). Using the established secret keys, the entities in the network communicate securely. The data gathered securely by the gateway nodes is then used to form various types of blocks (private, public or consortium) at the edge server(s) based on application types in the generic IoT environment. The created blocks are mined by the edge servers in order to add them in the blockchain center. A detailed security analysis including the formal security has revealed that the proposed CBACS-EIoT is robust against various potential attacks needed in the IoT environment. To further strengthen the security, the simulation based formal security verification on CBACS-EIoT has been carried out to exhibit that CBACS-EIoT is secure against passive and active attacks. Finally, a meticulous comparative performance analysis shows that CBACS-EIoT offers superior security and supports more functionality features, and also

provides less communication and computational overheads as compared to existing relevant schemes.

7.2 Open research challenges

In this section, we mention few challenging research problems that can be targeted to enhance the security in IoT-related applications.

7.2.1 Post-quantum access control

In an IoT environment, the communicated messages transmitted among the entities have the sensitive information. Thus, maintaining message integrity and data privacy become the challenging task due to resource-limited of the IoT smart devices.

In recent years, several security protocols, including sauthentication [139, 171, 184, 222], access control [65, 86, 128, 194] and key agreement [48, 76, 97, 195] are suggested in the literature. However, these security solutions are based on traditional number-theoretic assumptions. However, traditional cryptographic protocols dependent on the traditional number theory assumptions may not resist the quantum attacks, as stated by Shor [183]. To overcome these issues, a "post-quantum cryptography (known as quantum-resistant cryptography)" can be deployed in IoT environment that can provide security against both quantum and classical attacks [13]. Towards this goal, the "lattice-based cryptography (LBC)" can serve as a possible solution [46, 127, 129] due to the following major advantages:

- The computational hard problems like "integer factorization problem (IFP)" and "discrete logarithm problem (DLP)" may not be able to resist the quantum attacks, whereas there are no quantum algorithms till date that can solve the lattice based security protocols in polynomial time.
- The cryptographic protocols that are dependent on the hardness of "average case" lattice challenges can be further reduced to the hardness of "worst case" lattice problems as well. Most public key-based cryptosystems suffer from such a problem.
- Most operations in LBC are linear, and hence, the LBC framework will enjoy computational efficiency over traditional public-key based cryptosystems.

Because the IoT devices are resource constrained in nature, they have less memory and limited battery power. We need then design the security protocols that should be efficient enough so

that they can properly function in the network. Therefore, the lattice based security schemes should be adopted to the IoT devices in order to provide superior security as compared to traditional public-key schemes.

7.2.2 AI/ML-based Big data analytics

In an IoT environment, a huge amount of data is produce through various real-time applications. Thus, there is a great need for Big data analytics for the gathered data. Towards this, the machine learning (ML) which is a branch of artificial intelligence (AI) can provide "an intelligence service based on the data as well as algorithms by learning and improving its accuracy". Whether the collected data is genuine (authentic) or a noisy data, the ML techniques offer more accurate results on predictions. In case the data is poisonous, it may lead to wrong predictions [26, 53, 144, 190]. As a result, it would be interesting if the ML can be used for the malicious behavior of the deployed IoT devices in the network. In addition, the ML techniques could be useful for detection of attacks, data security and privacy, and malware analysis.

Bibliography

[1] Delegated Proof of Stake (DPoS). https://tokens-economy.gitbook.io/consensus/chain-based-proof-of-stake/delegated-proof-of-stake-dpos. Accessed on August 2023.

[2] Secure Hash Standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. Accessed on January 2021.

[3] Cisco Global Cloud Index: Forecast and Methodology, 2014. 2014–2019, Cisco, San Jose, CA, USA, White Paper.

[4] Blockchain Tutorial-Learn Blockchain Technology from Scratch, 2018. https://data-flair.training/blogs/blockchain-tutorial/. Accessed on October 2021.

[5] MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library, 2020. Accessed on April 2020.

[6] Raspberry Pi 3 Model B+, 2020. Accessed on April 2020.

[7] Consortium Blockchain, 2022. https://learn.bybit.com/glossary/definition-consortium-blockchain/. Accessed on March 2023.

[8] What's the Difference Between a Raspberry Pi and a Computer?, 2022. https://raspberrytips.com/difference-raspberry-pi-computer/. Accessed on October 2022.

[9] D. Abbasinezhad-Mood and M. Nikooghadam. Design and hardware implementation of a security-enhanced elliptic curve cryptography based lightweight authentication scheme for smart grid communications. *Future Generation Computer Systems*, 84:47 – 57, 2018.

[10] M. Abdalla, P. A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In *8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science*, volume 3386, pages 65–84, Les Diablerets, Switzerland, 2005.

[11] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K. K. R. Choo, and A. Y. Zomaya. Blockchain for smart communities: Applications, challenges and opportunities. *Journal of Network and Computer Applications*, 144:13 – 48, 2019.

[12] N. Z. Aitzhan and D. Svetinovic. Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams. *IEEE Transactions on Dependable and Secure Computing*, 15(5):840–852, 2018.

[13] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, 2020. https://doi.org/10.6028/NIST.IR.8309. Accessed on October 2021.

[14] M. Amoon, T. Altameem, and A. Altameem. RRAC: Role based reputed access control method for mitigating malicious impact in intelligent IoT platforms. *Computer Communications*, 151:238–246, 2020.

[15] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100:143 – 174, 2019.

[16] M. S. Arbabi, C. Lal, N. R. Veeraragavan, D. Marijan, J. F. Nygård, and R. Vitenberg. A Survey on Blockchain for Healthcare: Challenges, Benefits, and Future Directions. *IEEE Communications Surveys & Tutorials*, 25(1):386–424, 2023.

[17] AVISPA. Automated Validation of Internet Security Protocols and Applications, 2021. http://www.avispa-project.org/. Accessed on January 2021.

[18] AVISPA. SPAN, the Security Protocol ANimator for AVISPA, 2021. http://www.avispa-project.org/. Accessed on January 2021.

[19] H. Azimy and A. Ghorbani. Competitive Selfish Mining. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–8, 2019.

[20] M. F. Aziz, A. N. Khan, J. Shuja, I. A. Khan, F. G. Khan, and A. u. R. Khan. A lightweight and compromise-resilient authentication scheme for IoTs. *Transactions on Emerging Telecommunications Technologies*, 33(3):e3813, 2022.

[21] P. Bagchi, R. Maheshwari, B. Bera, A. K. Das, Y. Park, P. Lorenz, and D. K. Y. Yau. Public Blockchain-Envisioned Security Scheme Using Post Quantum Lattice-Based Aggregate Signature for Internet of Drones Applications. *IEEE Transactions on Vehicular Technology*, 72(8):10393–10408, 2023.

[22] E. Barker. Recommendation for Key Management. Special Publication 800-57 Part 1 Rev. 4, NIST, 01/2016, 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf. Accessed on March 2021.

[23] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 139–155, Bruges, Belgium, 2000.

[24] B. Bera, D. Chattaraj, and A. K. Das. Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment. *Computer Communications*, 153:229 – 249, 2020.

[25] B. Bera, A. K. Das, and S. K. Das. Search on Encrypted COVID-19 Healthcare Data in Blockchain-Assisted Distributed Cloud Storage. *IEEE Internet of Things Magazine*, 4(4):127–132, 2021.

[26] B. Bera, A. K. Das, M. S. Obaidat, P. Vijayakumar, K.-F. Hsiao, and Y. Park. AI-Enabled Blockchain-Based Access Control for Malicious Attacks Detection and Mitigation in IoE. *IEEE Consumer Electronics Magazine*, 10(5):82–92, 2021.

[27] B. Bera, A. K. Das, and A. K. Sutrala. Private blockchain-based access control mechanism for unauthorized UAV detection and mitigation in Internet of Drones environment. *Computer Communications*, 166:91–109, 2021.

[28] B. Bera, S. Saha, A. K. Das, N. Kumar, P. Lorenz, and M. Alazab. 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, 2020.

[29] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos. Designing Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System. *IEEE Internet of Things Journal*, 8(7):5744–5761, 2021.

[30] B. Bera, M. Wazid, A. K. Das, and J. J. P. C. Rodrigues. Securing Internet of Drones Networks Using AI-Envisioned Smart-Contract-Based Blockchain. *IEEE Internet of Things Magazine*, 4(4):68–73, 2021.

[31] J. B. Bernabe, J. L. Hernandez-Ramos, and A. F. S. Gomez. Holistic Privacy-Preserving Identity Management System for the Internet of Things. *Mobile Information Systems*, 2017:1–20, 2017. Article ID 6384186, DOI: 10.1155/2017/6384186.

[32] B. Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016.

[33] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly Secure Key Distribution for Dynamic Conferences. *Information and Computation*, 146(1):1–23, 1998.

[34] A. Braeken, P. Porambage, M. Stojmenovic, and L. Lambrinos. eDAAAS: Efficient distributed anonymous authentication and access in smart homes. *International Journal of Distributed Sensor Networks*, 12(12):1–11, 2016.

[35] J. Camhi. Former Cisco CEO John Chambers Predicts 500 Billion Connected Devices by 2025, 2015. Bus. Insider, New York, NY, USA.

[36] R. Canetti. Introduction to Cryptography. Lecture 9 - Symmetric Encryption, 2008. http://www.cs.tau.ac.il/~canetti/f08-materials/scribe9.pdf. Accessed on June 2020.

[37] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, pages 337–351, Amsterdam, The Netherlands, 2002.

[38] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002.

[39] S. Challa, M. Wazid, A. K. Das, N. Kumar, A. G. Reddy, E. Yoon, and K. Yoo. Secure Signature-Based Authenticated Key Establishment Scheme for Future IoT Applications. *IEEE Access*, 5:3028–3043, 2017.

[40] V. Chamola, V. Hassija, V. Gupta, and M. Guizani. 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, 2020.

[41] A. C. F. Chan and J. Zhou. Cyber-Physical Device Authentication for the Smart Grid Electric Vehicle Ecosystem. *IEEE Journal on Selected Areas in Communications*, 32(7):1509–1517, 2014.

[42] C. C. Chang and H. D. Le. A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. *IEEE Transactions on Wireless Communications*, 15(1):357–366, 2016.

[43] D. Chattaraj, B. Bera, A. K. Das, S. Saha, P. Lorenz, and Y. Park. Block-CLAP: Blockchain-Assisted Certificateless Key Agreement Protocol for Internet of Vehicles in Smart Transportation. *IEEE Transactions on Vehicular Technology*, 70(8):8092–8107, 2021.

[44] S. Chatterjee, A. K. Das, and J. K. Sing. An Enhanced Access Control Scheme in Wireless Sensor Networks. *Ad Hoc & Sensor Wireless Networks*, 21(1-2):121–149, 2014.

[45] S. Chatterjee, A. K. Das, and J. K. Sing. A novel and efficient user access control scheme for wireless body area sensor networks. *Journal of King Saud University - Computer and Information Sciences*, 26(2):181–201, 2014.

[46] R. Chaudhary, G. S. Aujla, N. Kumar, and S. Zeadally. Lattice-Based Public Key Cryptosystem for Internet of Things Environment: Challenges and Solutions. *IEEE Internet of Things Journal*, 6(3):4897–4909, 2019.

[47] S. A. Chaudhry, H. Alhakami, A. Baz, and F. Al-Turjman. Securing Demand Response Management: A Certificate-Based Access Control in Smart Grid Edge Computing Infrastructure. *IEEE Access*, 8:101235–101243, 2020.

[48] S. A. Chaudhry, M. S. Farash, N. Kumar, and M. H. Alsharif. PFLUA-DIoT: A Pairing Free Lightweight and Unlinkable User Access Control Scheme for Distributed IoT Environments. *IEEE Systems Journal*, 16(1):309–316, 2022.

[49] T. W. Chim, S. M. Yiu, V. O. K. Li, L. C. K. Hui, and J. Zhong. PRGA: Privacy-Preserving Recording amp; Gateway-Assisted Authentication of Power Usage Information for Smart Grid. *IEEE Transactions on Dependable and Secure Computing*, 12(1):85–97, 2015.

[50] C. J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification*, pages 414–418, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[51] K. A. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, 151:147–157, 2019.

[52] A. K. Das. A Secure User Anonymity-Preserving Three-Factor Remote User Authentication Scheme for the Telecare Medicine Information Systems. *Journal of Medical Systems*, 39(3):30, 2015.

[53] A. K. Das, B. Bera, and D. Giri. AI and Blockchain-Based Cloud-Assisted Secure Vaccine Distribution and Tracking in IoMT-Enabled COVID-19 Environment. *IEEE Internet of Things Magazine*, 4(2):26–32, 2021.

[54] A. K. Das, B. Bera, S. Saha, N. Kumar, I. You, and H.-C. Chao. AI-Envisioned Blockchain-Enabled Signature-Based Key Management Scheme for Industrial Cyber-Physical Systems. *IEEE Internet of Things Journal*, 9(9):6374–6388, 2022.

[55] A. K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, and X. Huang. Provably secure user authentication and key agreement scheme for wireless sensor networks. *Security and Communication Networks*, 9(16):3670–3687, 2016.

[56] A. K. Das, N. R. Paul, and L. Tripathy. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences*, 209:80–92, 2012. https://doi.org/10.1016/j.ins.2012.04.036.

[57] A. K. Das and I. Sengupta. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials. pages 9–16, Bangalore, India, 2008. 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08).

[58] A. K. Das, M. Wazid, N. Kumar, M. K. Khan, K.-K. R. Choo, and Y. Park. Design of Secure and Lightweight Authentication Protocol for Wearable Devices Environment. *IEEE Journal of Biomedical and Health Informatics*, 22(4):1310–1322, 2018.

[59] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal*, 5(6):4900–4913, 2018.

[60] A. K. Das, S. Zeadally, and D. He. Taxonomy and analysis of security protocols for Internet of Things. *Future Generation Comp. Syst.*, 89:110–125, 2018.

[61] A. K. Das, S. Zeadally, and M. Wazid. Lightweight authentication protocols for wearable devices. *Computers & Electrical Engineering*, 63:196–208, 2017.

[62] M. Davidson and T. Diamond. On the Profitability of Selfish Mining Against Multiple Difficulty Adjustment Algorithms. Cryptology ePrint Archive, Report 2020/094, 2020. https://ia.cr/2020/094.

[63] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, and A. Kashif Bashir. A survey of security and privacy issues in the Internet of Things from the layered context. *Transactions on Emerging Telecommunications Technologies*, 33(6):e3935, 2022.

[64] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[65] S. Ding, J. Cao, C. Li, K. Fan, and H. Li. A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT. *IEEE Access*, 7:38431–38441, 2019.

[66] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD. In D. Gollmann, editor, *Fast Software Encryption*, pages 71–82, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[67] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[68] J. R. Douceur. The Sybil Attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS'02), Lecture Notes in Computer Science (LNCS), Springer-verlag*, volume 2429, pages 251–260, Cambridge, MA, USA, 2002.

[69] M. Du, X. Ma, Z. Zhang, X. Wang, and Q. Chen. A review on consensus algorithm of blockchain. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC'17)*, pages 2567–2572, Banff, AB, Canada, 2017.

[70] S. K. Dwivedi, R. Amin, A. K. Das, M. T. Leung, K.-K. R. Choo, and S. Vollala. Blockchain-based vehicular ad-hoc networks: A comprehensive survey. *Ad Hoc Networks*, 137:102980, 2022.

[71] S. K. Dwivedi, R. Amin, S. Vollala, and A. K. Das. Design of Blockchain and ECC based Robust and Efficient Batch Authentication Protocol for Vehicular Ad-Hoc Networks. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[72] M. A. El-Zawawy, A. Brighente, and M. Conti. Authenticating Drone-Assisted Internet of Vehicles Using Elliptic Curve Cryptography and Blockchain. *IEEE Transactions on Network and Service Management*, 20(2):1775–1789, 2023.

[73] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, J. Rodriguez, A. Bicaku, S. Maksuti, M. G. Tauber, C. Schmittner, and J. Bastos. A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment. *IEEE Internet of Things Journal*, 6(1):288–296, 2019.

[74] D. Evans. The Internet of Things: How the next evolution of the Internet is changing everything, 2011. Cisco, San Jose, CA, USA, White Paper.

[75] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[76] D. Fang, Y. Qian, and R. Q. Hu. A Flexible and Efficient Authentication and Secure Data Transmission Scheme for IoT Applications. *IEEE Internet of Things Journal*, 7(4):3474–3484, 2020.

[77] C. Feng, B. Liu, Z. Guo, K. Yu, Z. Qin, and K.-K. R. Choo. Blockchain-Based Cross-Domain Authentication for Intelligent 5G-Enabled Internet of Drones. *IEEE Internet of Things Journal*, 9(8):6224–6238, 2022.

[78] H. Finney. RPOW - Reusable Proofs of Work, 2004. https://nakamotoinstitute.org/rpow/. Accessed on August 2023.

[79] M. Fiore and M. Mongiello. Blockchain Technology to Support Agri-Food Supply Chains: A Comprehensive Review. *IEEE Access*, 11:75311–75324, 2023.

[80] J. Frankenfield. Blockchain-as-a-Service (BaaS) Meaning and Major Players, 2021. https://www.investopedia.com/terms/b/blockchainasaservice-baas.asp. Accessed on June 2021.

[81] J. Frankenfield. Merkle Tree in Blockchain: What it is and How it Works, 2021. https://www.investopedia.com/terms/m/merkle-tree.asp. Accessed on March 2022.

[82] K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang. Permissioned Blockchain and Edge Computing Empowered Privacy-Preserving Smart Grid Networks. *IEEE Internet of Things Journal*, 6(5):7992–8004, 2019.

[83] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma. TrustAccess: A Trustworthy Secure Ciphertext-Policy and Attribute Hiding Access Control Scheme Based on Blockchain. *IEEE Transactions on Vehicular Technology*, 69(6):5784–5798, 2020.

[84] S. Garg, K. Kaur, G. Kaddoum, and K. R. Choo. Toward Secure and Provable Authentication for Internet of Things: Realizing Industry 4.0. *IEEE Internet of Things Journal*, 7(5):4598–4606, 2020.

[85] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[86] B. Gupta and M. Quamara. An identity based access control and mutual authentication framework for distributed cloud computing services in IoT environment using smart cards. *Procedia Computer Science*, 132:189–197, 2018.

[87] M. Gupta, F. M. Awaysheh, J. Benson, M. Alazab, F. Patwa, and R. Sandhu. An Attribute-Based Access Control for Cloud Enabled Industrial Smart Vehicles. *IEEE Transactions on Industrial Informatics*, 17(6):4288–4297, 2021.

[88] S. Haber and W. S. Stornetta. How to time-stamp a digital document. In *Advances in Cryptology-CRYPTO' 90*, pages 437–455, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[89] Z. Han, X. Li, G. Xu, N. Xiong, E. Merlo, and E. Stroulia. An Effective Evolutionary Analysis Scheme for Industrial Software Access Control Models. *IEEE Transactions on Industrial Informatics*, 16(2):1024–1034, 2020.

[90] X. Hao, W. Ren, K.-K. R. Choo, and N. N. Xiong. A Self-Trading and Authenticated Roaming Scheme Based on Blockchain for Smart Grids. *IEEE Transactions on Industrial Informatics*, 18(6):4097–4106, 2022.

[91] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen. Distributed Access Control with Privacy Support in Wireless Sensor Networks. *IEEE Transactions on Wirless Communications*, 10(10):3473–3481, 2011.

[92] D. He, H. Wang, M. K. Khan, and L. Wang. Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography. *IET Communications*, 10(14):1795–1802, 2016.

[93] Y. Hu, A. Perrig, and D. Johnson. Pachet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of IEEE International Conference on Computer and Communications (INFOCOM'03)*, volume 3, pages 1976–1986, San Francisco, CA, USA, 2003.

[94] H. F. Huang. A novel access control protocol for secure sensor networks. *Computer Standards & Interfaces*, 31:272–276, 2009.

[95] H. F. Huang. A New Design of Access Control in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2011, 2011. Article ID 412146, 7 pages, DOI: 10.1155/2011/412146.

[96] H. F. Huang and K. C. Liu. A New Dynamic Access Control in Wireless Sensor Networks. pages 901–906, Yilan, Taiwan, 2008. IEEE Asia-Pacific Services Computing Conference (APSCC'08).

[97] U. Iqbal and A. H. Mir. Secure and scalable access control protocol for IoT environment. *Internet of Things*, 12:100291, 2020.

[98] L. Ismail and H. Materwala. A Review of Blockchain Architecture and Consensus Protocols: Use Cases, Challenges, and Solutions. *Symmetry*, 11(10):1198, 2019.

[99] S. Jang, D. Lim, J. Kang, and I. Joe. An Efficient Device Authentication Protocol Without Certification Authority for Internet of Things. *Wireless Personal Communications*, 91(4):1681–1695, 2016.

[100] S. Jangirala, A. K. Das, and A. V. Vasilakos. 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, 2020.

[101] N. Jansma and B. Arrendondo. Performance Comparison of Elliptic Curve and RSA Digital Signatures. http://nicj.net/files/performance_comparison_of_elliptic_curve_and_rsa_digital_signatures.pdf. Accessed on March 2022.

[102] Q. Jiang, S. Zeadally, J. Ma, and D. He. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access*, 5:3376–3392, 2017.

[103] H. J. Jo, I. S. Kim, and D. H. Lee. Efficient and Privacy-Preserving Metering Protocols for Smart Grid Systems. *IEEE Transactions on Smart Grid*, 7(3):1732–1742, 2016.

[104] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[105] I. A. Kamil and S. O. Ogundoyin. EPDAS: Efficient privacy-preserving data analysis scheme for smart grid network. *Journal of King Saud University - Computer and Information Sciences*, 33(2):208–217, 2021.

[106] N. Khalil, M. R. Abid, D. Benhaddou, and M. Gerndt. Wireless sensors networks for Internet of Things. In *IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, Singapore, 2014.

[107] K. Khullar. Implementing PBFT in Blockchain), 2019. Accessed on July 2020.

[108] H. S. Kim and S. W. Lee. Enhanced novel access control protocol over wireless sensor networks. *IEEE Transactions on Consumer Electronics*, 55(2):492–498, 2009.

[109] S. K. Kim and J. H. Huh. A Study on the Improvement of Smart Grid Security Performance and Blockchain Smart Grid Perspective. *Energies*, 11(8):1–22, 2018.

[110] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[111] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[112] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *25th Annual International Cryptology Conference (CRYPTO'05)*, pages 546–566, Santa Barbara, California, USA, 2005. Springer.

[113] N. Kumar, G. S. Aujla, A. K. Das, and M. Conti. ECCAuth: A Secure Authentication Protocol for Demand Response Management in a Smart Grid System. *IEEE Transactions on Industrial Informatics*, 15(12):6572–6582, 2019.

[114] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[115] X. H. Le, S. Lee, I. Butun, M. Khalid, R. Sankar, M. Kim, M. Han, Y.-K. Lee, and H. Lee. An Energy-Efficient Access Control Scheme for Wireless Sensor Networks based on Elliptic Curve Cryptography. *Journal of Communications and Networks*, 11(6):599–606, 2009.

[116] J. Lee, M. Azamfar, and J. Singh. A blockchain enabled Cyber-Physical System architecture for Industry 4.0 manufacturing systems. *Manufacturing Letters*, 20:34–39, 2019.

[117] J. Leng, D. Yan, Q. Liu, K. Xu, J. L. Zhao, R. Shi, L. Wei, D. Zhang, and X. Chen. ManuChain: Combining Permissioned Blockchain With a Holistic Optimization Model as Bi-Level Intel-

ligence for Smart Manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):182–192, 2020.

[118] J. Leng, S. Ye, M. Zhou, J. L. Zhao, Q. Liu, W. Guo, W. Cao, and L. Fu. Blockchain-Secured Smart Manufacturing in Industry 4.0: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1):237–252, 2021.

[119] J. Leng, X. Zhu, Z. Huang, K. Xu, Z. Liu, Q. Liu, and X. Chen. ManuChain II: Blockchained Smart Contract System as the Digital Twin of Decentralized Autonomous Manufacturing Toward Resilience in Industry 5.0. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(8):4715–4728, 2023.

[120] F. Li, Y. Han, and C. Jin. Practical access control for sensor networks in the context of the Internet of Things. *Computer Communications*, 89-90:154–164, 2016.

[121] F. Li, J. Hong, and A. A. Omala. Efficient certificateless access control for industrial Internet of Things. *Future Generation Computer Systems*, 76:285–292, 2017.

[122] H. Li, R. Lu, L. Zhou, B. Yang, and X. Shen. An Efficient Merkle-Tree-Based Authentication Scheme for Smart Grid. *IEEE Systems Journal*, 8(2):655–663, 2014.

[123] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari. A Robust ECC-Based Provable Secure Authentication Protocol With Privacy Preserving for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 14(8):3599–3609, 2018.

[124] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K. R. Choo. A Robust and Energy Efficient Authentication Protocol for Industrial Internet of Things. *IEEE Internet of Things Journal*, 5(3):1606–1615, 2018.

[125] C. Lin, D. He, X. Huang, K. K. R. Choo, and A. V. Vasilakos. BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of Network and Computer Applications*, 116:42–52, 2018.

[126] Y. Liu, Q. Lu, S. Chen, Q. Qu, H. O'Connor, K.-K. Raymond Choo, and H. Zhang. Capability-based IoT access control using blockchain. *Digital Communications and Networks*, 7(4):463–469, 2021.

[127] Z. Liu, K.-K. R. Choo, and J. Grossschadl. Securing Edge Devices in the Post-Quantum Internet of Things Using Lattice-Based Cryptography. *IEEE Communications Magazine*, 56(2):158–162, 2018.

[128] Z. Liu, C. Guo, and B. Wang. A Physically Secure, Lightweight Three-Factor and Anonymous User Authentication Protocol for IoT. *IEEE Access*, 8:195914–195928, 2020.

[129] X. Lu, W. Yin, Q. Wen, K. Liang, L. Chen, and J. Chen. Message Integration Authentication in the Internet-of-Things via Lattice-Based Batch Signatures. *Sensors*, 18, 2018.

[130] Y. Lu, X. Tang, L. Liu, F. R. Yu, and S. Dustdar. Speeding at the Edge: An Efficient and Secure Redactable Blockchain for IoT-Based Smart Grid Systems. *IEEE Internet of Things Journal*, 10(14):12886–12897, 2023.

[131] M. Luo, Y. Luo, Y. Wan, and Z. Wang. Secure and efficient access control scheme for wireless sensor networks in the cross-domain context of the IoT. *Security and Communication Networks*, 2018:1–10, 2018.

[132] B. M. and K. S. G. An Authentication Protocol for Future Sensor Networks. *Sensors*, 17, 2017.

[133] K. Mahmood, S. A. Chaudhry, H. Naqvi, S. Kumari, X. Li, and A. K. Sangaiah. An elliptic curve cryptography based lightweight authentication scheme for smart grid communication. *Future Generation Computer Systems*, 81:557–565, 2018.

[134] K. Mahmood, X. Li, S. A. Chaudhry, H. Naqvi, S. Kumari, A. K. Sangaiah, and J. J. P. C. Rodrigues. Pairing based anonymous and secure key agreement protocol for smart grid edge computing infrastructure. *Future Generation Computer Systems*, 88:491 – 500, 2018.

[135] A. A. Mahmud and M. C. Morogan. Identity-based Authentication and Access Control in Wireless Sensor Networks. *International Journal of Computer Applications*, 41(13):18–24, 2012.

[136] N. Majedi, M. Naeem, and A. Anpalagan. Telecommunication integration in e-healthcare: technologies, applications and challenges. *Transactions on Emerging Telecommunications Technologies*, 27(6):775–789, 2016.

[137] S. Malani, J. Srinivas, A. K. Das, K. Srinathan, and M. Jo. Certificate-Based Anonymous Device Access Control Scheme for IoT Environment. *IEEE Internet of Things Journal*, 6(6):9762–9773, 2019.

[138] S. Mandal, B. Bera, A. K. Sutrala, A. K. Das, K. R. Choo, and Y. Park. Certificateless Signcryption-Based Three-Factor User Access Control Scheme for IoT Environment. *IEEE Internet of Things Journal*, 7(4):3184–3197, 2020.

[139] M. Masud, G. S. Gaba, K. Choudhary, M. S. Hossain, M. F. Alhamid, and G. Muhammad. Lightweight and Anonymity-Preserving User Authentication Scheme for IoT-Based Healthcare. *IEEE Internet of Things Journal*, 9(4):2649–2656, 2022.

[140] W. E. May. Secure Hash Standard, 2015. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. Accessed on August 2019.

[141] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, 2002.

[142] C. Miller. Stuart Haber and Scott Stornetta: How our timestamping mechanism was used in Bitcoin, 2021. https://coingeek.com/stuart-haber-and-scott-stornetta-how-our-timestamping-mechanism-was-used-in-\bitcoin-video/. Accessed on August 2023.

[143] A. K. Mishra, M. Wazid, D. P. Singh, A. K. Das, J. Singh, and A. V. Vasilakos. Secure Blockchain-Enabled Authentication Key Management Framework with Big Data Analytics for Drones in Networks Beyond 5G Applications. *Drones*, 7(8), 2023.

[144] A. Mitra, B. Bera, A. K. Das, S. S. Jamal, and I. You. Impact on blockchain-based AI/ML-enabled big data analytics for Cognitive Internet of Things environment. *Computer Communications*, 197:173–185, 2023.

[145] M. B. Mollah, J. Zhao, D. Niyato, K.-Y. Lam, X. Zhang, A. M. Y. M. Ghias, L. H. Koh, and L. Yang. Blockchain for Future Smart Grid: A Comprehensive Survey. *IEEE Internet of Things Journal*, 8(1):18–43, 2021.

[146] A. S. Musleh, G. Yao, and S. M. Muyeen. Blockchain Applications in Smart Grid–Review and Frameworks. *IEEE Access*, 7:86746–86757, 2019.

[147] B. Müßigmann, H. von der Gracht, and E. Hartmann. Blockchain Technology in Logistics and Supply Chain Management-A Bibliometric Literature Review From 2016 to January 2020. *IEEE Transactions on Engineering Management*, 67(4):988–1007, 2020.

[148] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008. https://downloads.coindesk.com/research/whitepapers/bitcoin.pdf. Accessed on March 2021.

[149] Y. Nakamura, Y. Zhang, M. Sasabe, and S. Kasahara. Capability-Based Access Control for the Internet of Things: An Ethereum Blockchain-Based Scheme. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Waikoloa, HI, USA, 2019.

[150] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: Analysis and defenses. In *Proceedings of third IEEE International Conference on Information Processing in Sensor Networks (IPSN'04)*, pages 259–268, Berkeley, California, USA, 2004.

[151] J. Ni, K. Zhang, X. Lin, and X. S. Shen. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. *IEEE Communications Surveys & Tutorials*, 20(1):601–628, 2018.

[152] H. Nicanfar, P. Jokar, K. Beznosov, and V. C. M. Leung. Efficient Authentication and Key Management Mechanisms for Smart Grid Communications. *IEEE Systems Journal*, 8(2):629–640, 2014.

[153] R. W. Nickalls. A new approach to solving the cubic: Cardan's solution revealed. *The Mathematical Gazette*, 77(480):354–359, 1993.

[154] O. Novo. Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.

[155] V. Odelu, A. K. Das, M. Khurram Khan, K. R. Choo, and M. Jo. Expressive CP-ABE Scheme for Mobile Devices in IoT Satisfying Constant-Size Keys and Ciphertexts. *IEEE Access*, 5:3273–3283, 2017.

[156] V. Odelu, A. K. Das, M. Wazid, and M. Conti. Provably Secure Authenticated Key Agreement Scheme for Smart Grid. *IEEE Transactions on Smart Grid*, 9(3):1900–1910, 2018.

[157] K. Park, J. Lee, A. K. Das, and Y. Park. BPPS:Blockchain-Enabled Privacy-Preserving Scheme for Demand-Response Management in Smart Grid Environments. *IEEE Transactions on Dependable and Secure Computing*, 20(2):1719–1729, 2023.

[158] B. Parno, A. Perrig, and V. Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *IEEE Symposium on Security and Privacy (S & P'05)*, pages 49–63, Oakland, California, USA, 2005.

[159] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918, 1978.

[160] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[161] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim. The Future of Healthcare Internet of Things: A Survey of Emerging Technologies. *IEEE Communications Surveys & Tutorials*, 22(2):1121–1167, 2020.

[162] M. Qi and J. Chen. Two-Pass Privacy Preserving Authenticated Key Agreement Scheme for Smart Grid. *IEEE Systems Journal*, 15(3):3201–3207, 2021.

[163] G. Raja, S. G. Senthivel, S. Balaganesh, B. R. Rajakumar, V. Ravichandran, and M. Guizani. MLB-IoD: Multi Layered Blockchain Assisted 6G Internet of Drones Ecosystem. *IEEE Transactions on Vehicular Technology*, 72(2):2511–2520, 2023.

[164] S. Ramzan, A. Aqdus, V. Ravi, D. Koundal, R. Amin, and M. A. Al Ghamdi. Healthcare Applications Using Blockchain Technology: Motivations and Challenges. *IEEE Transactions on Engineering Management*, 70(8):2874–2890, 2023.

[165] P. M. Rao, S. Jangirala, S. Pedada, A. K. Das, and Y. Park. Blockchain Integration for IoT-Enabled V2X Communications: A Comprehensive Survey, Security Issues and Challenges. *IEEE Access*, 11:54476–54494, 2023.

[166] P. P. Ray, D. Dash, K. Salah, and N. Kumar. Blockchain for IoT-Based Healthcare: Background, Consensus, Platforms, and Use Cases. *IEEE Systems Journal*, 15(1):85–94, 2021.

[167] S. Raza, L. Wallgren, and T. Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674, 2013.

[168] Y. Ren, F. Zhu, J. Qi, J. Wang, and A. K. Sangaiah. Identity Management and Access Control Based on Blockchain under Edge Computing for the Industrial Internet of Things. *Applied Sciences*, 9(10), 2019.

[169] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[170] Rivest, Ronald L. The MD4 Message Digest Algorithm. In *10th Annual International Cryptology Conference – Advances in Cryptology (CRYPTO'90)*, pages 303–311, Santa Barbara, CA, USA, 1991.

[171] M. Rodrigues, J. Amaro, F. S. Osório, and B. Kalinka. R. L. J. C. Authentication Methods for UAV Communication. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 1210–1215, Barcelona, Spain, 2019.

[172] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen. Exploring the Attack Surface of Blockchain: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 22(3):1977–2008, 2020.

[173] S. Saha, A. K. Sutrala, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. On the Design of Blockchain-Based Access Control Protocol for IoT-Enabled Healthcare Applications. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, Dublin, Ireland, 2020.

[174] N. Saxena, B. J. Choi, and R. Lu. Authentication and Authorization Scheme for Various User Roles and Devices in Smart Grid. *IEEE Transactions on Information Forensics and Security*, 11(5):907–921, 2016.

[175] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[176] D. Schwartz, N. Youngs, A. Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8), 2014.

[177] S. Sciancalepore, G. Piro, G. Boggia, and G. Bianchi. Public Key Authentication and Key Agreement in IoT Devices With Minimal Airtime Consumption. *IEEE Embedded Systems Letters*, 9(1):1–4, 2017.

[178] J. Sengupta, S. Ruj, and S. Das Bit. A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT. *Journal of Network and Computer Applications*, 149:102481, 2020.

[179] D. Shakhbulatov, J. Medina, Z. Dong, and R. Rojas-Cessa. How Blockchain Enhances Supply Chain Management: A Survey. *IEEE Open Journal of the Computer Society*, 1:230–249, 2020.

[180] D. Shanks. Class number, a theory of factorization, and genera. In *Proceedings of Symposia in Pure Mathematics*, volume 20, pages 41–440, 1971. https://doi.org/10.1090/pspum/020/0316385.

[181] Y. Sharaf-Dabbagh and W. Saad. On the authentication of devices in the Internet of things. In *17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–3, Coimbra, Portugal, 2016.

[182] J. Shen, S. Moh, and I. Chung. COMMENT: "Enhanced Novel Access Control Protocol over Wireless Sensor Networks". *IEEE Transactions on Consumer Electronics*, 56(3):2019–2021, 2010.

[183] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[184] M. Shuai, N. Yu, H. Wang, and L. Xiong. Anonymous authentication scheme for smart home environment with provable security. *Computers & Security*, 86:132–146, 2019.

[185] L. Soltanisehat, R. Alizadeh, H. Hao, and K.-K. R. Choo. Technical, Temporal, and Spatial Research Challenges and Opportunities in Blockchain-Based Healthcare: A Systematic Literature Review. *IEEE Transactions on Engineering Management*, 70(1):353–368, 2023.

[186] S. Son, J. Lee, Y. Park, Y. Park, and A. K. Das. Design of Blockchain-Based Lightweight V2I Handover Authentication Protocol for VANET. *IEEE Transactions on Network Science and Engineering*, 9(3):1346–1358, 2022.

[187] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng. A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes. *IEEE Internet of Things Journal*, 4(6):1844–1852, 2017.

[188] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment. *IEEE Transactions on Vehicular Technology*, 68(7):6903–6916, 2019.

[189] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. Cloud Centric Authentication for Wearable Healthcare Monitoring System. *IEEE Transactions on Dependable and Secure Computing*, 17(5):942–956, 2020.

[190] J. Srinivas, A. K. Das, M. Wazid, and A. V. Vasilakos. Designing Secure User Authentication Protocol for Big Data Collection in IoT-Based Intelligent Transportation System. *IEEE Internet of Things Journal*, 8(9):7727–7744, 2021.

[191] V. Srivastava, S. K. Debnath, B. Bera, A. K. Das, Y. Park, and P. Lorenz. Blockchain-Envisioned Provably Secure Multivariate Identity-Based Multi-Signature Scheme for Internet of Vehicles Environment. *IEEE Transactions on Vehicular Technology*, 71(9):9853–9867, 2022.

[192] W. Stallings. *Cryptography and Network Security: Principles and Practice, International Edition: Principles and Practice*. Pearson Upper Saddle River, 2014. ISBN: 0273793764.

[193] W. Su and A. Q. Huang. *The Energy Internet*. Sawston, U.K., Woodhead Publishing, 2018.

[194] B. H. Taher, S. Jiang, A. A. Yassin, and H. Lu. Low-Overhead Remote User Authentication Protocol for IoT Based on a Fuzzy Extractor and Feature Extraction. *IEEE Access*, 7:148950–148966, 2019.

[195] M. Tanveer, A. U. Khan, N. Kumar, A. Naushad, and S. A. Chaudhry. A Robust Access Control Protocol for the Smart Grid Systems. *IEEE Internet of Things Journal*, 9(9):6855–6865, 2022.

[196] Y. Teng, L. Li, L. Song, F. R. Yu, and V. C. M. Leung. Profit Maximizing Smart Manufacturing Over AI-Enabled Configurable Blockchains. *IEEE Internet of Things Journal*, 9(1):346–358, 2022.

[197] S. Thapliyal, S. Singh, M. Wazid, D. Singh, and A. K. Das. Design of blockchain-enabled secure smart health monitoring system and its testbed implementation. *Cyber Security and Applications*, 1:100020, 2023.

[198] H. Tian, X. Li, H. Quan, C.-C. Chang, and T. Baker. A Lightweight Attribute-Based Access Control Scheme for Intelligent Transportation System With Full Privacy Protection. *IEEE Sensors Journal*, 21(14):15793–15806, 2021.

[199] J. Tsai and N. Lo. Secure Anonymous Key Distribution Scheme for Smart Grid. *IEEE Transactions on Smart Grid*, 7(2):906–914, 2016.

[200] Y. P. Tsang, K. L. Choy, C. H. Wu, G. T. S. Ho, and H. Y. Lam. Blockchain-Driven IoT for Food Traceability With an Integrated Consensus Mechanism. *IEEE Access*, 7:129000–129017, 2019.

[201] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo. Efficient byzantine fault-tolerance. *IEEE Transactions on Computers*, 62(1):16–30, 2011.

[202] L. Vigano. Automated Security Protocol Analysis With the AVISPA Tool. *Electronic Notes in Theoretical Computer Science*, 155:61 – 86, 2006.

[203] D. von Oheimb. The high-level protocol specification language hlpsl developed in the eu project avispa. pages 1–17, Frauenchiemsee, Germany, 2005. Proceedings of 3rd APPSEM II (Applied Semantics II) Workshop (APPSEM'05).

[204] D. Wang, D. He, P. Wang, and C. H. Chu. Anonymous Two-Factor Authentication in Distributed Systems: Certain Goals Are Beyond Attainment. *IEEE Transactions on Dependable and Secure Computing*, 12(4):428–442, 2015.

[205] H. Wang, B. Sheng, and Q. Li. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks*, 1(3-4):127–137, 2006.

[206] J. Wang, L. Wu, K.-K. R. Choo, and D. He. Blockchain-Based Anonymous Authentication With Key Management for Smart Grid Edge Computing Infrastructure. *IEEE Transactions on Industrial Informatics*, 16(3):1984–1992, 2020.

[207] N. Wang, X. Zhou, X. Lu, Z. Guan, L. Wu, X. Du, and M. Guizani. When Energy Trading Meets Blockchain in Electrical Power System: The State of the Art. *Applied Sciences*, 9(8), 2019.

[208] X. Wang, G. Yu, R. P. Liu, J. Zhang, Q. Wu, S. W. Su, Y. He, Z. Zhang, L. Yu, T. Liu, W. Zhang, P. Loneragan, E. Dutkiewicz, E. Poole, and N. Paton. Blockchain-Enabled Fish Provenance and Quality Tracking System. *IEEE Internet of Things Journal*, 9(11):8130–8142, 2022.

[209] X. Wang, P. Zeng, N. Patterson, F. Jiang, and R. Doss. An Improved Authentication Scheme for Internet of Vehicles Based on Blockchain Technology. *IEEE Access*, 7:45061–45072, 2019.

[210] Y. Wang, Z. Su, N. Zhang, J. Chen, X. Sun, Z. Ye, and Z. Zhou. SPDS: A Secure and Auditable Private Data Sharing Scheme for Smart Grid Based on Blockchain. *IEEE Transactions on Industrial Informatics*, 17(11):7688–7699, 2021.

[211] M. Wazid, P. Bagga, A. K. Das, S. Shetty, J. J. P. C. Rodrigues, and Y. Park. AKM-IoV: Authenticated Key Management Protocol in Fog Computing-Based Internet of Vehicles Deployment. *IEEE Internet of Things Journal*, 6(5):8804–8817, 2019.

[212] M. Wazid, B. Bera, A. K. Das, S. Garg, D. Niyato, and M. S. Hossain. Secure Communication Framework for Blockchain-Based Internet of Drones-Enabled Aerial Computing Deployment. *IEEE Internet of Things Magazine*, 4(3):120–126, 2021.

[213] M. Wazid, B. Bera, A. K. Das, S. P. Mohanty, and M. Jo. Fortifying Smart Transportation Security Through Public Blockchain. *IEEE Internet of Things Journal*, 9(17):16532–16545, 2022.

[214] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. Rodrigues. Authentication in cloud-driven IoT-based big data environment: Survey and outlook. *Journal of Systems Architecture*, 97:185–196, 2019.

[215] M. Wazid, A. K. Das, N. Kumar, M. Conti, and A. V. Vasilakos. A Novel Authentication and Key Agreement Scheme for Implantable Medical Devices Deployment. *IEEE Journal of Biomedical and Health Informatics*, 22(4):1299–1309, 2018.

[216] M. Wazid, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. Secure Three-Factor User Authentication Scheme for Renewable-Energy-Based Smart Grid Environment. *IEEE Transactions on Industrial Informatics*, 13(6):3144–3153, 2017.

[217] M. Wazid, A. K. Das, S. Kumari, X. Li, and F. Wu. Design of an efficient and provably secure anonymity preserving three-factor user authentication and key agreement scheme for TMIS. *Security and Communication Networks*, 9(13):1983–2001, 2016.

[218] M. Wazid, A. K. Das, V. Odelu, N. Kumar, M. Conti, and M. Jo. Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks. *IEEE Internet of Things Journal*, 5(1):269–282, 2018.

[219] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo. Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment. *IEEE Transactions on Dependable and Secure Computing*, 17(2):391–406, 2020.

[220] K. E. Wegrzyn and E. Wang. Types of Blockchain: Public, Private, or Something in Between, 2021. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between. Accessed on October 2021.

[221] A. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[222] F. Wu, X. Li, L. Xu, P. Vijayakumar, and N. Kumar. A Novel Three-Factor Authentication Protocol for Wireless Sensor Networks with IoT Notion. *IEEE Systems Journal*, 15(1):1120–1129, 2021.

[223] H. Wu, S. Jiang, and J. Cao. High-Efficiency Blockchain-Based Supply Chain Traceability. *IEEE Transactions on Intelligent Transportation Systems*, 24(4):3748–3758, 2023.

[224] L. Wu, J. Wang, K. R. Choo, and D. He. Secure Key Agreement and Key Protection for Mobile Device User Authentication. *IEEE Transactions on Information Forensics and Security*, 14(2):319–330, 2019.

[225] R. Xu, J. Joshi, and P. Krishnamurthy. An Integrated Privacy Preserving Attribute-Based Access Control Framework Supporting Secure Deduplication. *IEEE Transactions on Dependable and Secure Computing*, 18(2):706–721, 2021.

[226] J. Xue, C. Xu, and Y. Zhang. Private Blockchain-Based Secure Access Control for Smart Home Systems. *KSII Transactions on Internet and Information Systems*, 12(12):6057–6078, 2018.

[227] Y. Yang. A Vision towards Pervasive Edge Computing. In *22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'19)*, page 1, Miami Beach, FL, USA, 2019.

[228] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian. Resource Trading in Blockchain-Based Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 15(6):3602–3609, 2019.

[229] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari. Enabling Drones in the Internet of Things With Decentralized Blockchain-Based Security. *IEEE Internet of Things Journal*, 8(8):6406–6415, 2021.

[230] G. Yu, X. Zha, X. Wang, W. Ni, K. Yu, P. Yu, J. A. Zhang, R. P. Liu, and Y. J. Guo. Enabling Attribute Revocation for Fine-Grained Access Control in Blockchain-IoT Systems. *IEEE Transactions on Engineering Management*, 67(4):1213–1230, 2020.

[231] S. Yu, J. Lee, A. K. Sutrala, A. K. Das, and Y. Park. LAKA-UAV: Lightweight authentication and key agreement scheme for cloud-assisted Unmanned Aerial Vehicle using blockchain in flying ad-hoc networks. *Computer Networks*, 224:109612, 2023.

[232] A. Zahoor, K. Mahmood, S. Shamshad, M. A. Saleem, M. F. Ayub, M. Conti, and A. K. Das. An access control scheme in IoT-enabled Smart-Grid systems using blockchain and PUF. *Internet of Things*, 22:100708, 2023.

[233] S. Zeadally, A. K. Das, and N. Sklavos. Cryptographic technologies and protocol standards for Internet of Things. *Internet of Things*, 14:100075, 2021.

[234] P. Zeng, K.-K. Choo, and D.-Z. Sun. On the Security of an Enhanced Novel Access Control Protocol for Wireless Sensor Networks. *IEEE Transactions on Consumer Electronics*, 56(2):566–569, 2010.

[235] X. Zeng, G. Xu, X. Zheng, Y. Xiang, and W. Zhou. E-AUA: An Efficient Anonymous User Authentication Protocol for Mobile IoT. *IEEE Internet of Things Journal*, 6(2):1506–1519, 2019.

[236] C. Zhang, G. Zhou, H. Li, and Y. Cao. Manufacturing Blockchain of Things for the Configuration of a Data- and Knowledge-Driven Digital Twin Manufacturing Cell. *IEEE Internet of Things Journal*, 7(12):11884–11894, 2020.

[237] C. Zhang, G. Zhou, J. Li, F. Chang, K. Ding, and D. Ma. A multi-access edge computing enabled framework for the construction of a knowledge-sharing intelligent machine tool swarm in Industry 4.0. *Journal of Manufacturing Systems*, 66:56–70, 2023.

[238] C. Zhang, L. Zhu, C. Xu, K. Sharif, R. Lu, and Y. Chen. APPB: Anti-Counterfeiting and Privacy-Preserving Blockchain-Based Vehicle Supply Chains. *IEEE Transactions on Vehicular Technology*, 71(12):13152–13164, 2022.

[239] H. Zhang, J. Wang, and Y. Ding. Blockchain-based decentralized and secure keyless signature scheme for smart grid. *Energy*, 180:955–967, 2019.

[240] L. Zhang, Y. Cui, and Y. Mu. Improving Security and Privacy Attribute Based Data Sharing in Cloud Computing. *IEEE Systems Journal*, 14(1):387–397, 2020.

[241] R. Zhang, R. Xue, and L. Liu. Security and Privacy for Healthcare Blockchains. *IEEE Transactions on Services Computing*, 15(6):3668–3686, 2022.

[242] X. Zhang, P. Sun, J. Xu, X. Wang, J. Yu, Z. Zhao, and Y. Dong. Blockchain-Based Safety Management System for the Grain Supply Chain. *IEEE Access*, 8:36398–36410, 2020.

[243] Y. Zhang, X. Xu, A. Liu, Q. Lu, L. Xu, and F. Tao. Blockchain-Based Trust Mechanism for IoT-Based Smart Manufacturing System. *IEEE Transactions on Computational Social Systems*, 6(6):1386–1394, 2019.

[244] Y. Zhang, M. Yutaka, M. Sasabe, and S. Kasahara. Attribute-Based Access Control for Smart Cities: A Smart-Contract-Driven Framework. *IEEE Internet of Things Journal*, 8(8):6372–6384, 2021.

[245] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[246] Y. Zhou, Y. Guan, Z. Zhang, and F. Li. A Blockchain-Based Access Control Scheme for Smart Grids. *IACR Cryptology ePrint Archive*, 2019:880, 2019.

[247] Y. Zhou, Y. Zhang, and Y. Fang. Access control in wireless sensor networks. *Ad Hoc Networks*, 5:3–13, 2007.

Publications from this thesis work

Journal Papers

J1. Sourav Saha, Basudeb Bera, Ashok Kumar Das, Neeraj Kumar, SK Hafizul Islam, and YoungHo Park. "Private Blockchain Envisioned Access Control System for Securing Industrial IoT-Based Pervasive Edge Computing," in ***IEEE Access***, Vol. 11, pp. 130206-130229, 2023, DOI: 10.1109/ACCESS.2023.3333441. (2022 SCI Impact Factor: 3.9)

J2. Basudeb Bera, Sourav Saha, Ashok Kumar Das, and Athanasios V. Vasilakos. "Designing Blockchain-Based Access Control Protocol in IoT-Enabled Smart-Grid System," in ***IEEE Internet of Things Journal***, Vol. 8, No. 7, pp. 5744-5761, April 2021, DOI: 10.1109/JIOT.2020.3030308. (2022 SCI Impact Factor: 10.6)

J3. Sourav Saha, Durbadal Chattaraj, Basudeb Bera, and Ashok Kumar Das. "Consortium blockchain-enabled access control mechanism in edge computing based generic IoT environment," in ***Transactions on Emerging Telecommunications Technologies (Wiley)***, Vol. 32, No. 6, pp. 1-34, Article No. e3995, 2021, DOI: 10.1002/ETT.3995. (2022 SCI Impact Factor: 3.6)

Other publications

Journal Papers

J1. Sourav Saha, Ashok Kumar Das, Mohammad Wazid, YoungHo Park, Sahil Garg, and Mubarak Alrashoud. "Smart Contract-Based Access Control Scheme for Blockchain Assisted 6G-Enabled IoT-Based Big Data Driven Healthcare Cyber Physical Systems," in ***IEEE Transactions on Consumer Electronics***, 2024, DOI: 10.1109/TCE.2024.3391667. (2022 SCI Impact Factor: 4.3)

J2. Anusha Vangala, Basudeb Bera, Sourav Saha, Ashok Kumar Das, Neeraj Kumar, and YoungHo Park. "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in Intelligent Transportation Systems," in ***IEEE Sensors Journal***, Vol. 21, No. 14, pp. 15824-15838, July 2021, DOI: 10.1109/JSEN.2020.3009382. (2022 SCI Impact Factor: 4.3)

J3. Anil Kumar Sutrala, Mohammad S. Obaidat, Sourav Saha, Ashok Kumar Das, Mamoun Alazab, and YoungHo Park. "Authenticated Key Agreement Scheme with User Anonymity and Untraceability for 5G-Enabled Softwarized Industrial Cyber-Physical Systems," in ***IEEE Transactions on Intelligent Transportation Systems***, Vol. 23, No. 3, pp. 2316-2330, March 2022, DOI: 10.1109/TITS.2021.3056704. (2022 SCI Impact Factor: 8.5)

J4. Durbadal Chattaraj, Basudeb Bera, Ashok Kumar Das, Sourav Saha, Pascal Lorenz, and YoungHo Park. "Block-CLAP: Blockchain-assisted Certificateless Key Agreement Protocol for Internet of Vehicles in Smart Transportation," in ***IEEE Transactions on Vehicular Technology***, Vol. 70, No. 8, pp. 8092-8107, Aug. 2021, DOI: 10.1109/TVT.2021.3091163. (2022 SCI Impact Factor: 6.8)

J5. Ashok Kumar Das, Basudeb Bera, Sourav Saha, Neeraj Kumar, Ilsun You, and Han-Chieh Chao. "AI-Envisioned Blockchain-Enabled Signature-Based Key Management Scheme for Industrial Cyber-Physical Systems," in ***IEEE Internet of Things Journal***, Vol. 9, No. 9, pp. 6374-6388, May 2021, DOI: 10.1109/JIOT.2021.3109314. (2022 SCI Impact Factor: 10.6)

J6. Dharminder Dharminder, Ashok Kumar Das, Sourav Saha, Basudeb Bera, and Athanasios V. Vasilakos. "Post-Quantum Secure Identity-Based Encryption Scheme using Random Integer Lattices for IoT-enabled AI Applications," in ***Security and Communication Networks (Wiley-Hindawi)***, Vol. 2022, Article ID: 5498058, 2022, https://doi.org/10.1155/2022/5498058. (2021 SCI Impact Factor: 1.968)

Conference Papers

C1. Sourav Saha, Anil Kumar Sutrala, Ashok Kumar Das, Neeraj Kumar, and Joel J. P. C. Rodrigues. "On the Design of Blockchain-Based Access Control Protocol for IoT-Enabled Healthcare Applications," in ***54th IEEE International Conference on Communications (ICC 2020)***, Dublin, Ireland, 2020, pp. 1-6, DOI: 10.1109/ICC40277.2020.9148915.

C2. Durbadal Chattaraj, Sourav Saha, Basudeb Bera, and Ashok Kumar Das. "On the Design of Blockchain-Based Access Control Scheme for Software Defined Networks," in ***IEEE International Conference on Computer Communications (INFOCOM) Workshops: BlockSecSDN: Blockchain for Secure Software Defined Networking in Smart Communities***, Toronto, ON, Canada, 2020, pp. 237-242, DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162669.

C3. Sourav Saha, Ashok Kumar Das, and Debasis Giri. "Private Blockchain Enabled Security Framework for IoT-Based Healthcare System," in ***9th International Conference on Mathematics and Computing (ICMC 2023)***, Lecture Notes in Networks and Systems, Vol 697, pp. 99-115, Springer, Singapore, January 6-8, 2023, Goa, India, DOI: 10.1007/978-981-99-3080-7_8.