# Service Selection Middleware using Blockchain for IoT Applications

by

Syed Muhammad Danish

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, AUGUST 18, 2022

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

**BOARD OF EXAMINERS**

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

M. Kaiwen Zhang, Thesis supervisor
Department of software engineering and IT, École de technologie supérieure

M. Hans-Arno Jacobsen, Thesis Co-Supervisor
Department of Electrical & Computer Engineering, University of Toronto

M. Matthew Toews, Chair, Board of Examiners
Département de génie des systèmes, École de technologie supérieure

M. Abdelhakim Senhaji Hafid, External Examiner
Department of Computer Science and Operations Research University of Montreal

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "AUGUST 5, 2022"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

## ACKNOWLEDGEMENTS

My first and foremost gratitude belongs to ALLAH, the Almighty, who has blessed me with so many blessings throughout my life.

I would like to thank my supervisor, Prof. Kaiwen Zhang, for his guidance throughout my PhD journey. He has always supported me through thick and thin. He has led me in the correct direction over the past few years not only with his knowledge and experience, but also with the attitude for high quality research. I must say, I am very fortunate to have him as my advisor. Thank you very much for your enormous support and patience during my PhD.

I would also like to thank my co-supervisor Prof. Hans-Arno Jacobsen for their kind and patient supervision. My academic journey would not have been so rewarding without their kindness and wisdom. He continuously supported my PhD study and related research with his motivation, patience, sense of confidence in me and immense knowledge.

Afterwards, I would like to thank my near and dear friend and colleague named Nouman Ashraf for his help and support during my PhD journey. I would also like to thank my dear friends, inside and outside Canada, for their unwavering support and encouragement. I also thank all the lab mates for the fruitful discussions and making this journey fun and interesting. I am blessed to have such amazing friends in my life.

Lastly, thanks to my family for supporting me during this process. My parents have always encouraged me to pursue my goals.

**Les services intergiciels pour des applications IdO-Chaînes de blocs**

Syed Muhammad Danish

## RÉSUMÉ

Dans le monde d'aujourd'hui, l'Internet des objets (IdO) a inauguré une révolution sans précédent dans les technologies de l'information et est utilisé dans de nombreux secteurs, notamment la vente au détail, la fabrication et la santé. Chaque application IdO individuelle est différente en ce qui concerne les exigences fonctionnelles et non fonctionnelles, définies par le propriétaire de l'application ou l'industrie. La sélection des fournisseurs de services dans des environnements décentralisés est devenue un problème critique pour différentes applications IdO, en particulier lorsque les fournisseurs de services ont des fonctionnalités similaires mais les qualités des services (QDS) différente. Par conséquent, sélectionner un service de haute qualité qui répond le mieux aux exigences des applications IdO parmi une longue liste de services fonctionnellement équivalents est une tâche difficile. En outre, il est difficile de sécuriser les informations privées et critiques de l'application IdO, car le partage des paramètres de service avec des solutions tierces ou même avec le fournisseur de services lui-même peut créer de graves problèmes de sécurité et de confidentialité, car les exigences de service pourraient contiennent des informations sensibles concernant l'application, qui pourraient être utilisées à mauvais escient ou vendues par le tiers ou le fournisseur de services. Par conséquent, des mécanismes de sélection de services décentralisés sont nécessaires pour sélectionner le meilleur fournisseur de services pour les applications IdO, tout en garantissant la sécurité et la confidentialité de bout en bout des données sensibles associées.

Dans cette thèse, nous considérons d'abord un problème de sélection de stockage de données IdO, qui se concentre sur une conception de l'intergiciel pour la sélection de stockage de données intelligente basée sur la blockchain pour les applications IdO à grande échelle. Le cadre proposé étend l'architecture cloud IdO actuelle et considère les solutions de stockage d'égal à égal et basées sur la blockchain ainsi que les technologies cloud et multi-cloud. Nous modélisons le problème de sélection de stockage de données IdO comme un problème d'optimisation de décision et proposons deux algorithmes en temps polynomial comme solution. Nous proposons également un protocole vérifiable de sélection de stockage basé sur la blockchain, qui permet aux applications IoT ainsi qu'aux technologies de stockage de vérifier l'exactitude de la décision de placement des données prise par la conception de l'intergiciel sans avoir besoin d'un tiers de confiance. Enfin, nous proposons une stratégie de maintenance intelligente, qui prend en compte et apprend les fonctionnalités évolutives dynamiques des exigences de service des applications IoT pour optimiser la complexité de calcul ainsi que le stockage de la blockchain et les frais généraux des transactions dans la conception de l'intergiciel.

Le deuxième aspect des travaux de la thèse est le développement d'un protocole de sélection de bornes de recharge (CS) efficace et sécurisé basé sur la blockchain pour les réseaux de recharge de véhicules électriques (VE). Nous proposons une architecture décentralisée de recharge de VE basée sur la blockchain et modélisons théoriquement un problème d'optimisation de décision

décentralisée, qui élimine le besoin de tout tiers de confiance et permet aux VE et aux CS de communiquer de manière décentralisée, via les contrats intelligents, fonctionnant sur un réseau blockchain. Il permet également aux EV de sélectionner un CS et d'effectuer une réservation à distance avec un CS sans partager aucune information privée avec les CS ou toute entité de gestion centrale.

La troisième partie de cette thèse prolonge le deuxième aspect de cette thèse, et concerne les problèmes de sécurité et de confidentialité posés par la possibilité de lier les adresses publiques de la blockchain avec l'identité physique du propriétaire du VE. Pour résoudre ce problème, nous proposons un protocole de réservation de CS de bout en bout basé sur la blockchain, qui permet aux VE de réserver un créneau de charge au CS sélectionné en privé, sans partager leurs informations privées et en dissociant sa véritable identité de l'adresse blockchain, préservant ainsi la confidentialité du véhicule électrique. Comme les informations fournies par CS ne sont pas fiables, nous proposons un protocole de vérification des informations CS basé sur SMC qui permet aux véhicules électriques de vérifier de manière collaborative la disponibilité des emplacements de recharge à partir du CS en partageant en toute sécurité les réservations dans un environnement non fiable. Enfin, nous proposons une conception de contrat intelligente basée sur des protocoles de dépôt à verrouillage temporel, qui agrège le solde de telle sorte qu'aucun CS ne soit en mesure de lier les paiements de service de recharge aux adresses de blockchain des utilisateurs de VE, tout en garantissant que les CS reçoivent le paiement intégral pour les services qu'ils fournissent.

La contribution majeure de la thèse est de fournir des mécanismes de sélection de service sécurisés et efficaces adaptés à différentes applications IoT avec différentes exigences de service et besoins de sécurité. Grâce à l'approche proposée, les consommateurs pourront sélectionner en toute sécurité un fournisseur de services approprié en fonction de leurs besoins. Les résultats expérimentaux ont montré que les approches proposées obtenaient de meilleures performances et une meilleure efficacité par rapport à l'état de l'art.

**Mots-clés:** Les Chaînes de blocs, L'Internet des objets, Les véhicules électriques,Sécurité et confidentialité, L'intergiciel

**Service Selection Middleware using Blockchain for IoT Applications**

Syed Muhammad Danish

## ABSTRACT

In today's world, the Internet of Things (IoT) has ushered in an unprecedented revolution in information technology and is used in many industries, including retail, manufacturing and healthcare. Each individual IoT application is different when it comes to functional and non-functional requirements, defined by the application owner or industry. Service provider selection in a decentralized environments have become a critical issue for different IoT applications, in particular when services providers having similar functionality but different Quality of Service (QoS). As a result, selecting a high quality service that best suits IoT application requirements from a large list of functionally equivalent services is a challenging task. Furthermore, it is challenging to secure the private and business-critical information of the IoT application, as the sharing of service parameters with third-party solutions or even with the service provider itself can create serious security and privacy concerns, since the service requirements could contain sensitive information regarding the application, that could be misused or sold by the third-party or service provider. Therefore, a decentralized service selection mechanisms are required to select the best service provider for IoT applications, while ensuring end-to-end security and privacy of the associated sensitive data.

In this thesis, we first consider an IoT data storage selection problem, which focuses on a middleware design for blockchain-based intelligent data storage selection for large-scale IoT applications. The proposed framework extends the current IoT cloud architecture and considers peer-to-peer (P2P) and blockchain-based storage solutions along with cloud and multi-cloud technologies. We model IoT data storage selection problem as a decision optimization problem and propose two polynomial-time algorithms as a solution. We also propose a verifiable blockchain-based storage selection protocol, which enables the IoT applications as well as the storage technologies to verify the correctness of the data placement decision made by the middleware design without the need of any trusted third party. Finally, we propose an intelligent maintenance strategy, which takes into account and learns the dynamically evolving features of the IoT applications service requirements to optimizes the computational complexity along with the blockchain storage and transactions overhead in the middleware design.

The second aspect of the work in the thesis is the development of a blockchain-based efficient and secure charging station (CS) selection protocol for electric vehicles (EV) charging networks. We propose a decentralized blockchain-based EV charging architecture and theoretically model a decentralized decision optimization problem, which eliminates the need of any trusted third party and enables the EVs and the CSs to communicate in a decentralized manner, through the smart contracts, running on a blockchain network. It also enables EVs to select a CS and make a remote reservation with a CS without sharing any private information with CSs or any central management entity.

The third part of this thesis extends the second aspect of this thesis, and concerns the security and privacy problems arisen by the linkability of the public blockchain addresses with the EV owner's physical identity. To solve this problem, we propose a blockchain-based end-to-end privacy-preserving CS reservation protocol, which enables EVs to reserve a charging slot at the selected CS privately, without sharing their private information and by dissociating its real identity from the blockchain address, thereby preserving the EV's privacy. As the information provided by CS cannot be trusted, we propose an SMC-based CS information verification protocol that allows EVs to collaboratively verify the availability of charging slots from the CS by securely sharing reservations in an untrusted environment. Finally, we propose a smart contract design based on time-lock deposit protocols, which aggregates the balance in such a way that no CS is able to link charging service payments with EV users' blockchain addresses, while still ensuring that CSs receive full payment for the services they provide.

The major contribution of the thesis is providing a secure and efficient service selection mechanisms tailored for different IoT applications with different service requirements and security needs. With the proposed approach consumers will be able to securely select a suitable service provider based on their requirements. Experimental results showed that proposed approaches achieved better performance and efficiency compared to state-of-the-art.

**Keywords:** Internet of things, Blockchains, Electric vehicles, Security and privacy, Middleware

## TABLE OF CONTENTS

Page

## LIST OF TABLES

Page

## LIST OF FIGURES

Page

## LIST OF ALGORITHMS

Page

## LIST OF ABBREVIATIONS

| | |
|---|---|
| IoT | Internet of Things |
| IoEV | Internet of Electric Vehicles |
| QoS | Quality of Service |
| EV | Electric Vehicle |
| CS | Charging Station |
| SLA | Service Level Agreement |
| OCPP | Open Charge Point Protocol |
| P2P | Peer-to-Peer |
| SMC | Secure Multiparty Computation |
| PoW | Proof of Work |
| PoS | Proof of Stake |
| PBFT | Practical Byzantine Fault Tolerance |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| SL | Supervised Learning |
| UL | Unsupervised Learning |
| RNN | Recurrent Neural Network |
| LSTM | Long Short Term Memory |
| NN | Neural Network |

| | |
|---|---|
| ReLU | Rectified Linear Unit |
| TEE | Trusted Execution Environment |
| SGX | Software Guard Extensions |
| TPA | Third Party Auditor |
| UFL | Uncapacitated Facility Location |
| DP | Dynamic Programming |
| GS | Greedy Style |
| BW | Bandwidth |
| FLP | Facility Location Problem |
| LR | Linear Regression |
| kNN | K Nearest Neighbors |
| RF | Random Forest |
| Gauss | Gaussian |
| SVM | Support Vector Machine |
| BP | Back Propagation |
| RMSE | Root Mean Square Error |
| MAPE | Mean Absolute Percentage Error |
| ITS | Intelligent Transportation Systems |
| DLT | Distributed Ledger Technology |
| RSU | Road Side Unit |

| | |
|---|---|
| VANET | Vehicular Adhoc Network |
| GPS | Global Positioning System |
| SoC | State of Charge |
| API | Application Programming Interface |
| MITM | Man-in-the-Middle |
| ECC | Elliptic Curve Cryptography |
| TCB | Trusted Computing Base |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| FIFO | First In First Out |
| IAS | Intel Attestation Service |
| LCM | Least Common Multiple |

## LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

| | |
|---|---|
| $D$ | Number of IoT applications |
| $J$ | Number of storage technologies |
| $F_{d,j}$ | Data generated by storage technology j |
| $\mathcal{Y}$ | Decision variable |
| $C_{BW}$ | Bandwidth cost |
| $C_{CS}$ | Storage and retrieval cost |
| $C_{LT}$ | Latency cost |
| $C_{AV}$ | Availability cost |
| $C_{PR}$ | Privacy cost |
| $\mathcal{B}$ | Bandwidth budget |
| $\mathcal{A}$ | Availability threshold |
| $\mathcal{L}$ | Latency threshold |
| $\mathcal{M}$ | Monetary budget |
| $\gamma$ | Number of service requirements |
| $i_n$ | Input gate |
| $f_n$ | Forget gate |
| $o_n$ | Output gate |
| $v_i$ | Data placement decision |
| $T$ | Probability threshold |

| | |
|---|---|
| $d(y_t)$ | Amount of data at fog node |
| $A_r^{total}$ | Total aggregation rate |
| $d_{total}$ | Total amount of data generated |
| $Cert_{CS_m}$ | Digital certificate of EV |
| $Rep_{CS_m}$ | Reputation of CS |
| $C_{mon}$ | Monetary cost |
| $C_{tra}$ | Travelling cost |
| $C_{char}$ | Charging cost |
| $C_{wait}$ | Waiting time cost |
| $\mathcal{T}_{trip}^{max}$ | Travelling time threshold |
| $\mathcal{T}_{char}^{max}$ | Monetary cost threshold |
| $\mathcal{T}_{wait}^{max}$ | Waiting time threshold |
| $pk_{EV}$ | Public key of EV |
| $sk_{EV}$ | Private key of EV |
| $\mathcal{R}$ | Set of RSUs |
| $\mathcal{G}$ | Set of EVs |
| $pk_{CS}$ | Public key of CS |
| $sk_{CS}$ | Private key of CS |
| $pk_{RSU}^i$ | Public key of RSU |
| $sk_{RSU}^i$ | Private key of RSU |

| | |
|---|---|
| $h()$ | Hash function |
| $E()$ | Encryption function |
| $\mathbb{Z}_n^*$ | Set of integers |

# INTRODUCTION

IoT (Internet of Things) is a notion that sees the world as a connected smart environment where billions of physical objects are attached with sensors and actuators that can communicate and interact. In today's world, IoT has ushered in an unprecedented revolution in information technology (Nord, Koohang & Paliszkiewicz, 2019). The potential of this disruptive technology to improve lives, save time and money has caught the attention of the academy, society, and industry alike. (Lee & Lee, 2015). In the Internet of Things, people are able to interact with physical objects, such as cars, home appliances, and smartphones by sharing data, enabling more intelligent services to be developed, and improving the quality of life and comfort of users. (Abbate, Cesaroni, Cinici & Villari, 2019).

IoT devices possess limited computational power, memory, and/or energy availability, and its massive deployment resulted in the production of terabytes of heterogeneous data per day. IoT is used in many industries, including retail, manufacturing and healthcare, to enable real-time, informed decisions by gathering and processing data in real time (Le, Le Tuan & Tuan, 2019). One of the most notable examples of the Internet of Things in action is the Internet of Electric Vehicles (IoEV), which is a vast network involving vehicles, humans, sensors, charging stations, etc. These entities may interact and communicate with each other to collect and disseminate information about vehicles and the environment (Bayram & Papapanagiotou, 2014). The primary goal of the IoEV is to reduce accidents, improve traffic safety and efficiency, and enhance driving experience (Fraiji, Azzouz, Trojet & Saidane, 2018).

There are a wide range of IoT applications, from pollution monitoring to smart cities to digital health, and each individual IoT application is different when it comes to functional and non-functional requirements. A non-functional requirement is a parameter that relates to the quality of service (QoS), which is essential to meeting the needs of end users. Consumers as well as industrial IoT applications are currently served by different service providers on the market.

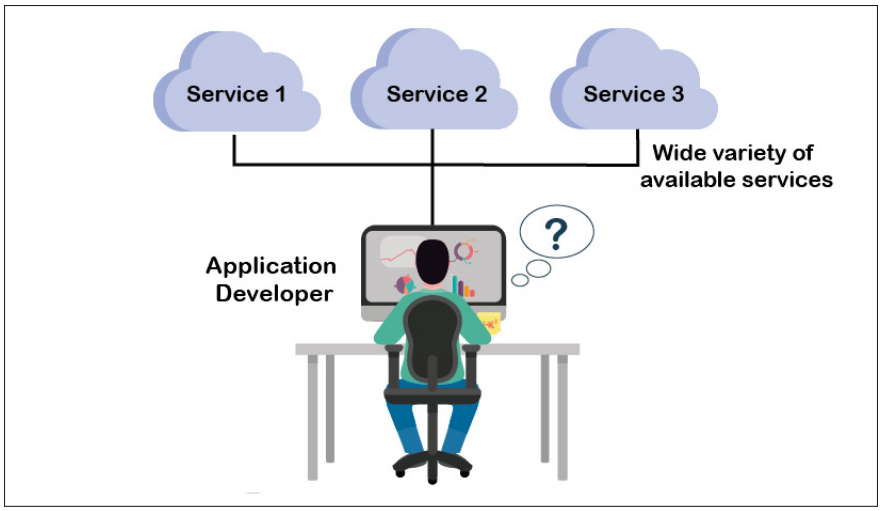



Figure 0.1 Service selection scenario

A few examples include the storage of IoT data for third parties, computing data, and buying and selling data. Additionally, although all service providers may offer the same functionality, they differ in terms of performance, security, price, etc. (Ruiz-Alvarez & Humphrey, 2011). Because there are a variety of different service providers, the selection of a particular service needs to be based on the needs and preferences of the end user (Xia, Chen, Bao, Wang & Yang, 2011; Abosaif & Hamza, 2020).

The art of selecting the most suitable service provider that fits the IoT application's needs has not been paid close attention given the recent advancement of IoT technology, which has resulted in many service providers entering the market with differentiated business models and service abstractions (Tuan, Thanh & Le Tuan, 2019). This becomes especially problematic when there are a lot of functionally equivalent services with different quality of service standards. Additionally, each IoT application has its own QoS parameters requirements, or different ways to express QoS parameters, which represents a significant challenge in selecting the right service.

This thesis is about investigating the challenges faced in service selection decisions, and developing solutions to enable efficient and secure service selection mechanisms for consumers as well as industrial IoT applications use cases.

**Challenges in Service Selection**

In some companies or organizations, making a good decision about the service they should hire is extremely important, since it affects their business in terms of future development, security, prices, etc. In the same vein, the IoT service selection process is also challenging, as it does not simply entail identifying and selecting the best service providers, but instead, the challenges associated with IoT service selection are as follows:

- In general, it may seem simple to identify and select the best service providers for an IoT application, but not so simple to model and implement, especially when each application has unique characteristics and needs. To make an informed decision, IoT applications' customized requirements and conditions must be considered in terms of security, reliability, availability, performance, and cost (Abosaif & Hamza, 2020). For example, industrial IoT applications may consider the overall performance provided by the service provider as well as some specific performance aspects, e.g., security and privacy of business information. On the other hand, a consumer IoT application will tend to take the cost/price of the service into account. Hence, an effective service selection strategy must consider the specific needs and conditions of various consumers.
- Considering the evolving nature of large-scale IoT applications and increasing number of competitive service providers, a highly scalable and reliable service selection solution is required. A service selection problem is usually modeled as an optimization problem, and the optimization problem can take a long time to solve if the input is too large or search space too large (Moghaddam & Davis, 2014). In some time-critical IoT applications, a long time to find a suitable service provider may not be practical, e.g., the selection of a storage provider for a large-scale IoT real-time streaming application should be made quickly (Barnaghi *et al.*,

2014). Hence, polynomial-time solutions to the service selection problem are needed, which can scale effectively to large numbers of IoT applications and service providers.

- Services are typically selected using broker solutions, which take service requirements from IoT applications and match them to QoS parameters from service providers (Papaioannou, Bonvin & Aberer, 2012; Chauhan, Pilli & Joshi, 2021). The sharing of service parameters with third-party solutions or even with the service provider itself can create serious security and privacy concerns, since the service requirements could contain private or business-critical information that could be misused or sold by the third-party or service provider (Squicciarini, Carminati & Karumanchi, 2011; Rahman, Khalil, Alabdulatif & Yi, 2019). So, it is necessary to protect the privacy needs of IoT application owners when selecting a service.
- As discussed above, an IoT application's security and privacy can be seriously compromised by third-party brokerage solutions. Apart from the misuse of critical personal information, future service providers or brokerage solutions may use this data to track the owner of an IoT application or its decisions and developments (Celik *et al.*, 2018; Ramnath, Javali, Narang, Mishra & Routray, 2017). For example, an IoEV application scenario might involve an electric vehicle (EV) sharing personal information, such as location and charging information, with a third party or a charging station (CS) that can be used in the future for vehicle location tracking as well as for predicting the EV user's habits, such as working hours, workplace, and other important patterns (Knirsch, Unterweger & Engel, 2018). Hence, it is imperative to ensure that no private information is shared or linked to a certain service selection request to ensure the long-term security and privacy of IoT applications.
- Lastly, we cannot trust IoT applications, service providers, and the organization responsible for making a service selection decision, so traceability and auditability are essential for making sure that the organization responsible for making the service selection decision is acting appropriately. Additionally, it is imperative to verify that the responsible entity has picked the best service by using the correct parameters during the service selection process.

Any entity wishing to verify the accuracy and validity of the service selection decision must have access to the QoS requirements of IoT applications, the parameters offered by service providers, and the associated service selection decision.

We propose the use of blockchain-based distributed ledger technology (DLT) (Danish *et al.*, 2020a) to address the above mentioned challenges in IoT service selection. Blockchain technology was introduced with Bitcoin (Nakamoto, 2008) to solve the double-spending problem and is widely adopted for a large number of crypto-currencies, such as Ethereum, Ripple, EOS, Zcash etc. (Zhang & Jacobsen, 2018). Blockchain, or distributed ledger, is a series of transaction records that are immutable, so if one of the records is modified, the rest of the peers will invalidate the transaction. One key emerging use case of blockchain technology involves "smart contracts" (Rathee *et al.*, 2019). Smart contract is a collection of code (its functions) and data (its state) that resides at a specific address on the blockchain. Through smart contract addresses, nodes on the blockchain network interact with smart contracts. The IoT has adopted this technology for the major problems of security, privacy, and provenance tracking related to the IoT based on its potential (Syed *et al.*, 2019).

As discussed previously, auditability and traceability is important to ensure trust and confidence while operating in an untrusted environment. The immutable and highly available transactions record in the blockchain network ensures traceability of user data and key communication among different entities. Similarly, blockchain eliminates the need of trusted third party, and the interacting parties can directly communicate with each others with the help of smart contracts, without sharing any private information thus, protecting user's and business's privacy. Moreover, the transparent and tamper-proof nature of blockchain network ensure secure trading among different parties without the need of trusted third party. Blockchain smart contracts can be used to enforce trading and payment logic, which can be executed automatically to ensure honest

behavior of the interacting parties. In addition, an automatic penalty logic can be programmed in the smart contracts to impose a penalty in case of deviation from the normal behavior.

**Research Motivation**

In this section, a few examples are introduced to motivate our work in practice.

**IoT supply chain application:** Consider an IoT supply chain application, where a large amount of data related to products and applications need to be stored. The data security and privacy assurance are crucial to the company's reputation as data processing and storage vulnerabilities can result in outside attacks and leaks. Choosing a storage service provider that has a very good reputation on data privacy and security is necessary if the supply chain company plans to move its work to the cloud to reduce daily costs. In this example, the factors other than security and privacy can also be considered, for instance, data availability and response time. In order to ensure fully functional supply chain process, the company wants to ensure that the data is available at all times with no risk of data loss. Similarly, response time can be a major concern for the supply chain application since data can be frequently accessed and corresponding encryption and decryption methods may be required. Finally, the company wants to reduce the overall price associated with the storage service provider along with the other requirements.

Therefore, a service selection approach must take into account all the service requirements of an IoT application along with the quality of service requirements of all the available service providers to choose the best one among them accurately. Moreover, the service requirements and service level agreements (SLA) associated with the storage service provider have to be maintained so that if the storage service provider fails to meet the application's requirements in the future, these SLAs can be traced and audited. According to the regulations, this information can later be used to issue penalties.

**Electric vehicle application:** Similar to the above example, in an IoEV application, the owner of an electric vehicle would like to select a charging station that is closest to where the vehicle is parked. Recent developments in CS deployment with competing energy companies in a liberalized market have led to increased dynamic pricing with respect to electricity load and available charging levels, e.g. fast vs. slow charging. Additionally, these CSs are located in different areas of a city in relation to the EV's current location, and the time needed to reach them varies with the current traffic and road conditions. The EV must wait at each CS before it is charged, so each CS has a waiting time associated with it. Normally, EV users' service requirements are usually elastic; for example, some prefer shorter waiting times at a lower charge but with more waiting and traveling time, while others prefer a lower charge but with a longer waiting period. Considering the diversity of CSs available, EV owners can pick the most appropriate CS based on their specific charging requirements, such as price, waiting time, charging time, and travel time.

Thus, an EV should not choose and engage a CS randomly, but instead assess its service requirements and match them with the services offered by CSs to select the right one. In addition, EV should not give its personal information to the CS or any entity in the network, and this service selection decision has to be made by EV itself in order to safeguard its private information, for instance, its current location, the state of its battery, etc. Last but not least, the reservation information should be stored so that it cannot be altered, and in case of a dispute, the stored information will be used to penalize the customer service as required by the regulations.

**Problem statement**

The goal of this thesis is to proposes an efficient and secure service selection mechanism for different IoT applications use cases for the purpose of improving the overall QoS, and end user comfort and security. We argue that enabling the owners to select a service provider for their IoT application can result in a more efficient and secure system as a whole, thus giving more

selection power to IoT applications and removing the need of trusted third parties. The premise of this thesis argues that giving the service selection power to individual IoT application will help them in meeting their specific QoS needs, instead of selecting a service provider randomly. Moreover, it will also help them in keeping their private information secure, thus enhancing the overall security.

A second objective is to investigate the blockchain integration to eliminate the need of trusted third party brokers, who make the service selection decision for IoT applications. Little work exist on the blockchain-based service selection mechanism to eliminate the need of trusted third parties, and to ensure end-to-end compelling security properties, e.g., traceability, auditability, accountability and data privacy protection, in blockchain-based IoT service selection design.

Concretely, this thesis studies the above research questions in three IoT application use-cases: large-scale IoT data storage selection (conference paper published IEEE Dapps (Danish & et al., 2020), and journal paper published in IEEE transactions on mobile computing (Danish, Zhang & Jacobsen, 2021)), charging station selection by electric vehicles (paper published in IEEE transactions on intelligent transportation systems (Danish, Zhang, Jacobsen, Ashraf & Qureshi, 2020c)), and privacy-preserving charging station selection, reservation and payment by electric vehicles (paper submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems)

**Large-scale IoT data storage selection:** Traditionally, IoT applications infrastructure relies only on a single cloud storage technology, which limits the possible service requirements of IoT applications. In addition, relying on a single cloud storage technology increases the risk of technology or vendor lock-in and places limits on the application and SLA requirements (Rafique & et al, 2017). Considering the large variety of cloud and decentralized storage options available and their underlying heterogeneous storage systems, as well as their different promised Service Level Agreement (SLA) guarantees (Rafique & et al, 2019), we develop and implement a middleware design, which is responsible for the selection of an effective storage

technology to fulfill the service requirements and parameters of a particular IoT application by jointly considering cloud and decentralized storage. We theoretically formulated a data storage selection problem as a decision optimization problem, which runs on the proposed middleware and finds the storage solution based on the captured cost functions of storage technologies and service requirements of IoT applications. We also integrate blockchain technology to ensure traceability, auditability and data placement decision correctness verifiability by IoT applications as well as storage technology. We performed experiments by considering cloud, multi-cloud and blockchain-based storage technologies and our results show that our middleware effectively reduces the price of IoT data storage while maintaining service requirements

**Charging station selection by electric vehicles:** Currently, industry standard protocols are enabling communication between EVs and CSs in the charging network, e.g., ISO 15118 and Open Charge Point Protocol (OCPP) (Antoun, Kabir, Moussa, Atallah & Assi, 2020). However, these industry standard protocols suffer from severe security and privacy threats. The ISO 15118 protocol involves a trusted intermediary mobility operator, which keeps the private information (EV identity, location, state of charge, charging parameters, availability and payment information) of EVs to authenticate and identify them (Eiza, Shi, Marnerides, Owens & Ni, 2018; Bao, Valev, Wagner & Schmeck, 2018). As the private information of the EVs is handled by a trusted centralized intermediary, the collection of such private data and tracking of an EV raise serious privacy and security concerns. To address these problems, we propose the use of blockchain technology, which enables the EVs to communicate with the CSs without sharing their private information. It aims to ensure the trust, privacy, auditability and traceability and enforces EVs and CSs to honestly honor the energy trading protocol. we also propose an efficient decentralized CS selection mechanism for the EVs, which allows EVs to select the CS themselves with high QoS and enhanced EV user comfort, while keeping EV's private information secure. Finally, we conduct an experimental evaluation and perform sensitivity analysis to demonstrate that our

proposed protocol is scalable and incurs significantly low blockchain transaction and storage overhead compared to state-of-the-art.

**Privacy-preserving charging station selection, reservation and payment:** Although, the previous work (Danish *et al.*, 2020c) solves the problem securing EV's private information however, at the CS premises, the linkability of the blockchain addresses to the physical identities can pose a serious risk to EV's privacy, as EV owners' charging histories and payment information are associated with their wallet address on the blockchain network. To solve the problem, we propose a blockchain-based private CS reservation protocol that allows EV owners to reserve a charging slot privately, without sharing their personal information or exposing their blockchain addresses at the CS. Unlike our previous work, the proposed solution ensures end-to-end privacy preservation for EV owners, i.e., the blockchain address of EV owner cannot be linked to its physical identity at the time of reservation and payment for the charging service. We conduct an experimental evaluation and perform sensitivity analysis to demonstrate that our solution ensures end-to-end EV owners' privacy with low blockchain transaction and computation overhead.

**Contributions**

In a nutshell, the core contribution of the thesis is to proposes an efficient and secure service selection mechanism for different IoT applications use cases with the specific needs and preferences defined by the end-users/businesses to improve overall QoS and protection of user's private information. In order to address the significant and challenging issues introduced above, this thesis makes contributions:

1. The first contribution to the project is BlockAIM, which focuses on a middleware design for blockchain-based intelligent data storage selection for large-scale IoT applications. We first proposed the framework, which extends the current IoT cloud architecture, and considers P2P and blockchain-based storage solutions along with cloud and multi-cloud technologies. We model IoT data storage selection problem as a decision optimization problem and

propose two polynomial-time algorithms as a solution. Then, we propose a verifiable blockchain-based storage selection protocol, which enables the IoT applications as well as the storage technologies to verify the correctness of the data placement decision made by the middleware design without the need of any trusted third party. Finally, we propose an intelligent maintenance strategy, which takes into account and learns the dynamically evolving features of the IoT applications service requirements to optimizes the computational complexity along with the blockchain storage and transactions overhead in the middleware design.

2. The second core contribution is BlockEV, which is a blockchain-based efficient and secure CS selection protocol for EV charging networks. First, we propose a decentralized blockchain-based EV charging architecture, which eliminates the need of any trusted third party and enables the EVs and the CSs to communicate in a decentralized manner. Then, we propose a blockchain-based EV charging protocol, which enables EVs to select a CS and make a remote reservation with a CS without the need of any trusted intermediary central management system. Finally, we theoretically model the decentralized CS selection mechanism as a decision optimization problem, where a EV selects a CS without sharing any private information.
3. An EV charging network reservation and payment scheme on the blockchain based on privacy-preserving CS would be the third contribution. First, we propose a blockchain-based end-to-end privacy-preserving CS reservation protocol, which enables EVs to reserve a charging slot at the selected CS privately, without sharing their private information and by dissociating its real identity from the blockchain address, thereby preserving the EV's privacy. Then, we propose an SMC-based CS information verification protocol that allows EVs to collaboratively verify the availability of charging slots from the CS by securely sharing reservations in an untrusted environment. Finally, we propose a smart contract design based on time-lock deposit protocols, which aggregates the balance in such a way

that no CS is able to link charging service payments with EV users' blockchain addresses, while still ensuring that CSs receive full payment for the services they provide.

**Thesis Organization**

The rest of this document is organized as follows. First, we provide the related work for each contribution in Chapter 1. Second, we provide in Chapter 2, a review of background knowledge, concepts, and techniques required for proper understanding of all aspects of this thesis. Then, each of the core sections focuses on one of the aforementioned contributions: Chapter 3 for blockchain-based intelligent data storage selection for IoT applications, Chapter 4 for blockchain-based charging station selection for EVs, and Chapter 5 for privacy-preserving CS reservation for EVs. In each section, we provide additional background material necessary for understanding that particular work. Each chapter also provides experimental evaluation. Finally, the conclusion chapter summarizes our findings and provides an outlook on current and future work.

# CHAPTER 1

# RELATED WORK

We now provide a detailed analysis of the state-of-the-art related work. We structure this chapter by listing the relevant literature for each of our contributions.

## 1.1 Blockchain-based middleware for iot data storage selection

This section reviews the state-of-the-art related studies on multi-cloud storage systems (Wu & et al, 2013; AlZain & et al, 2011; Dobre & et al, 2014; Cachin, Haas & Vukolic, 2010; Bowers & et al, 2009; Rafique & et al, 2019, 2017), blockchain-based IoT data storage (Shafagh & et al, 2017; Li *et al.*, 2018; Dorri & et al, 2017; Karlsson & et al, 2018) and draws a comparison between these studies and our work.

### 1.1.1 Blockchain-based storage

In recent times, Blockchain technology has been emerged as a promising solution to solve the IoT privacy and security issues. To store IoT data off-chain, a blockchain-based framework is proposed in (Li *et al.*, 2018). In the proposed blockchain-based framework, Distributed Hash Tables (DHTs) are employed to store large-scale IoT data: pointers to this data are written in the blockchain network. To store time series IoT data, authors in (Shafagh & et al, 2017) proposed a blockchain-based data storage framework. They also proposed an off-chain storage mechanism based on DHTs to store IoT data. Moreover, authors in (Dorri & et al, 2017) considered a smart home IoT application and proposed a blockchain-based architecture, which utilized cloud storage technologies to store IoT data from smart homes. Authors in (Karlsson & et al, 2018) proposed a partition-tolerant blockchain ,i.e., Vegvisir, which aims to target the IoT devices with limited power and limited network connectivity. In order to solve the problem of low storage capability of IoT devices, the authors integrated a support blockchain in their architecture to store the IoT data.

Compared to our work, a single storage solution has been considered in all these proposed blockchain-based frameworks for all the IoT applications. These frameworks lack auditable data provenance for large-scale IoT data and ignores distinct service requirements of different IoT applications. Moreover, all these works lack a traceable and accountable middleware design to dynamically assess and adapt the correct storage solution. Our proposed middleware solution BlockAM addresses all of the above issues.

### 1.1.2 Multi-cloud storage

In recent times, a lot of work has been done for multi-cloud storage system (Wu & et al, 2013; AlZain & et al, 2011; Dobre & et al, 2014; Cachin *et al.*, 2010; Bowers & et al, 2009), which considers the application service requirements in order to decide the storage solution. SPANStore (Wu & et al, 2013) minimizes the cost, while placing multiple replicas across multi-cloud to distribute the trust over multiple storage providers along with enhancing security and availability. In order to maximize the security of user data, authors in (AlZain & et al, 2011) propose a multi-cloud database model, i.e., MCDB. Hybris (Dobre & et al, 2014) maximizes the confidentiality of the customer's data by spreading the encrypted data of the customer over multi-clouds. However, these multi-cloud systems often target single objective optimization, i.e., every system focus on specific application requirement at a time rather than considering multiple requirements. Moreover, these multi-cloud systems do not consider recent decentralized storage technologies, which significantly differ from traditional cloud storage systems in term of performance and privacy. These multi-cloud systems focus on the dynamic behaviour of cloud systems and do not consider the dynamic behaviour of applications itself. Finally, these multi-cloud architectures do not provide traceability, accountability and auditability to clients and storage technologies.

However, there exists a small number of multi-cloud papers which do consider applications with multiple requirements optimization goal (Rafique & et al, 2019, 2017). In particular, (Rafique & et al, 2019) proposes a middleware design which only considers cloud storage technologies, while ignoring decentralized storage. Furthermore, the authors only consider the

dynamic behaviour of cloud storage, while ignoring the dynamic behaviour of IoT application service requirements. In addition, the solution lacks a theoretical formulation for data storage selection. Lastly, the proposed framework lacks traceability, accountability and middleware's decision correctness verifiability for the IoT applications as well as storage technologies. Our proposed middleware BlockAM addresses all the above issues.

### 1.1.3 Decentralized storage technologies

**Storj:** Storj (Wilkinson, Boshevski, Brandoff & Buterin, 2014) is a cryptocurrency and a decentralized storage technology. The Storj network consists of clients, storage nodes and satellites. Storj is a partially decentralized network in which satellite nodes control the storage nodes and clients. Storj provides end-to-end encryption to the client's data and use erasure coding to add redundancy to ensure high availability. Storj creates chunks of client's data and distribute these chunks to the storage nodes all around the world. In order to compete with cloud solutions, to increase the scalability and performance of the network, Storj does not employ blockchain and byzantine distributed consensus and everything is controlled by Satellites.

**Filecoin:** Filecoin (Psaras & Dias, 2020) is a decentralized storage network and works as an incentive layer over the InterPlanetary File System (IPFS) protocol. Filecoin operates using a blockchain with Filecoin tokens, which are paid to the miners in return of providing storage to clients. Similarly, clients use Filecoin to hire storage nodes to store their data. Unlike Storj, the Filecoin network works without a trusted third party. Storage nodes publish their storage capacity as well as the price on the Filecoin blockchain network and the interested client can bid on this information. Once the client and the storage node agree on a deal, this deal is then saved in the Filecoin blockchain. Filecoin also provides end-to-end data encryption and can distribute the data to multiple storage nodes. Filecoin network has two markets: the storage market and the retrieval market. In the storage market, the Filecoin network adds two transactions to the blockchain network: the storage information publication by storage node and the signed deal. The retrieval market works using the gossip protocol to retrieve the client's data.

**Safe Network:** The Safe network (Lambert & Bollen, 2014) aims to provide a decentralized peer-to-peer World Wide Web. Decentralized storage is therefore a major aspect of this network. Similar to Storj and Filecoin, Safe network also employs the free space of the storage nodes to store client data, using end-to-end encryption. The data is first converted to chunks and these chunks are then distributed to multiple storage nodes worldwide. Safe network has its own cryptocurrency called Safecoin, which is required to use applications on the decentralized Safe network. Safecoin does not use blockchain and has its own transaction verification mechanism to reach consensus. Safe network has different client managers which are chosen randomly to manage client data. Upon reaching consensus that a client has enough safecoins to save the data, these client managers will distribute the client data to multiple storage nodes.

Table 1.1 provides the comparison between cloud, Storj, Filecoin and Safe network in terms of decentralization, support for blockchain and smart contracts, storage price control, anonymity and the data location control.

Table 1.1 Comparison of storage technologies

| Storage | Decentralization | Smart Contract | Blockchain | Price Decision | Anonymity | Data Location |
|---|---|---|---|---|---|---|
| Cloud | Centralized | No | No | Cloud | No | Cloud server |
| Storj | Partially centralized | No | No | Storj | No | Multiple storage nodes worldwide |
| Filecoin | Fully decentralized | Yes | Yes | Storage nodes | No | One or more storage nodes |
| Safe network | Fully decentralized | No | No | Safe network | Yes | Multiple storage nodes worldwide |

## 1.2 Blockchain-based middleware for CS selection

In this section, we review the related studies based on EVs scheduling and selection mechanisms (Jin, Tang & Ghosh, 2013; Tsaousoglou, Steriotis & Varvarigos, 2019; Mukherjee & Gupta, 2014; He, Bai & Zhu, 2016; Nejad, Mashayekhy, Chinnam & Grosu, 2017; Yang & Yang, 2014; Goyal, Sharma, Vyas & Kumar, 2016; He, Zhu, Zhang & Zheng, 2018; Shi & Wong, 2011; Cao *et al.*, 2011; Patil & Kalkhambkar, 2019; Wen, Linde, Ropke, Mirchandani & Larsen; del Razo & , 2016; Schmidt, Saucke & Spengler, 2018; Sweda & Klabjan, 2012; Pourazarm, Cassandras & Malikopoulos, 2014; Said, Cherkaoui & Khoukhi, 2013; Yang, Cheng, Hsu, Gan & Lin, 2013; Bodet, Schülke, Erickson & Jabłonowski, 2012; Qin & Zhang, 2011;

Gusrialdi, Qu & Simaan, 2017) and blockchain-based EV frameworks (Knirsch *et al.*, 2018; Kirpes & Becker, 2018; Jin *et al.*, 2019; Kim *et al.*; Pustišek, Kos & Sedlar, 2016; Kamuni, Asfia, Sutavani, Sheikh & Patel, 2019; Su *et al.*, 2018; Jeong, Dao, Lee, Lee & Cho, 2018; Radi, Lasla, Bakiras & Mahmoud, 2019; Gao *et al.*, 2018; Samuel *et al.*, 2019), followed by a detailed comparison with existing studies.

### 1.2.1 Scheduling EV charging

In the literature, studies propose different methods for optimal scheduling of EVs at the CSs to optimize the overall cost of charging for EVs and CSs. In (Jin *et al.*, 2013; Goyal *et al.*, 2016) authors propose the scheduling of the EVs by jointly considering the EVs as well as aggregator's revenue. Authors in (Tsaousoglou *et al.*, 2019) provide a method for stochastic approximation of shadowed prices of CSs to schedule the price-elastic EVs. Work in (Mukherjee & Gupta, 2014) proposed a scheduling mechanism for EVs that maximize the number of EVs being charged at a time, while minimizing overall charging cost. In (He *et al.*, 2016; Yang & Yang, 2014; He *et al.*, 2018; Shi & Wong, 2011; Cao *et al.*, 2011; Patil & Kalkhambkar, 2019), authors proposed an optimal charging scheme to schedule the EVs based on their requirements to minimize the cost of EV owners. Work in (Nejad *et al.*, 2017) proposed an online scheduling and pricing mechanism for EV charging in an auction-based platform to incentivize both CSs and EVs. Several works have also been proposed to minimize the travelling time for the EVs. Authors in (Wen *et al.*), formulated the EVs scheduling problem to firstly minimize the number of vehicles needed to cover all the timetabled trips, and secondly to minimize the total traveling distance. In (del Razo & , 2016), authors propose a distributed scheduling approach based on A* algorithm and that makes use of charging station reservation system aimed at minimizing the total travel time for each EV. Authors in (Schmidt *et al.*, 2018) proposed a scheduling mechanism for police EVs, which aims to minimize the total cost, which consists of the cost of travel and the cost of delay. Authors in (Sweda & Klabjan, 2012) introduced a recharging plan for EVs to find a charging station with the shortest path based on the minimum travel time only. Similarly,

authors in (Pourazarm *et al.*, 2014) studied the routing problem for EVs in order to minimize the travelling time and recharging time.

Studies also propose different EVs scheduling methods to minimize the overall waiting time at the CSs. An algorithm of directing EV flows to charging stations is proposed in (Said *et al.*, 2013) to distribute the charging load and to minimize queuing time by having the EVs communicate with the transportation network. Authors in (Yang *et al.*, 2013) selects the best CS with minimum queuing time by estimating the queuing time of all the CSs. In (Bodet *et al.*, 2012), authors proposed a scheduling mechanism to optimize the CSs utilization by minimizing the EVs waiting time. Authors in (Qin & Zhang, 2011) investigated the EV charging scheduling activities by minimizing the waiting time at the CSs. In (Gusrialdi *et al.*, 2017), authors proposed a scheduling strategy, which allows the CSs coordination to minimize the waiting time at the CSs. However, in all these works the decision has been taken by the central aggregator and CSs excluding the EVs. Moreover, the decision for EVs is taken by communicating the information with central aggregator and CSs, which could lead to leakage of private information of EVs. To the best of our knowledge, no previous work consider the decentralized CS selection problem from EV's perspective without sharing its private information to other entities in electric vehicle charging network. Therefore, a decentralized mechanism for efficient selection of CSs by EVs is needed, where each EV is capable of selecting a CS in a distributed manner, based on its service requirements without sharing their private information to central aggregator and CSs.

### 1.2.2 Blockchain-based EV frameworks

Several works have been proposed to integrate blockchain with the EVs network. Authors in (Kirpes & Becker, 2018; Jeong *et al.*, 2018), proposed a blockchain and smart contract based payment transaction verification mechanism for EVs and CSs charging records on blockchain. Work in (Jin *et al.*, 2019) focuses on the allocation and trading of EV charging right among CSs through the blockchain network. Authors in (Kim *et al.*) proposed a blockchain-based EV charging system to improve security, efficiency and key management. A blockchain-based framework is proposed to optimally allocate CSs to EVs through smart contract infrastructure

(Pustišek *et al.*, 2016). In (Kamuni *et al.*, 2019), authors proposed blockchain-based framework to trade energy between EVs and CSs, while storing cost of CSs on the blockchain network. Authors proposed a blockchain-based dynamic optimal contract assignment and energy allocation algorithm for EVs charging scenario in (Su *et al.*, 2018). However, in all these works, there is no mechanism which provides security to EV's private information as the EVs are communicating with the central aggregator or CSs. Moreover, these blockchain systems incur high blockchain storage and transaction fee overheads.

Authors in (Radi *et al.*, 2019) proposed a blockchain-based energy trading mechanism for EVs along with anonymous payment mechanism however their work considers a consortium blockchain with an assumption of trusted third parties. In (Gao *et al.*, 2018), authors proposed a blockchain-based data sharing framework with anonymous payments however, their work is focused on private payment mechanism and ignores the selection of CSs by EVs. Work in (Samuel *et al.*, 2019) proposed a blockchain-based energy trading mechanism for EVs and CSs however, in their framework the EVs are sharing private information with a central aggregator which could lead to serious privacy concerns. To the best of our knowledge, only the work in (Knirsch *et al.*, 2018) is related to our proposed framework. Unlike (Knirsch *et al.*, 2018) our proposed framework not only ensures the same level of privacy for the EV users, but also ensures the availability of reserved time slot and trusted payment with selected CS. Work in (Knirsch *et al.*, 2018) lacks theoretical formulation of selection of CSs however, in this work we propose a mathematical formulation for efficient selection of CSs based on the EVs requirements. It should be noted that this work solves the CS selection problem for an individual EV in a decentralized manner and no EV scheduling is done in this paper. Furthermore, our proposed framework incurs less storage and transaction overhead in the blockchain network. Compared to the work in (Knirsch *et al.*, 2018), our proposed framework ensures the profitability of the CSs, since the selected CS will take part in the blockchain interaction with EV and other CSs do not need to spend money in blockchain transactions. Finally, in contrast to (Knirsch *et al.*, 2018), our work considers user based reputation mechanism for CSs, which is used as a feedback to select the future CS by EVs.

## 1.3 Privacy-preserving EV charging slot reservation and payment

Being related to the above work, the relevant literature is also similar, especially with regards to EV charging slot reservation. The work presented in Chapter 4 is also related to CS selection and reservation, but unlike the work in Chapter 4, its focus more on end-to-end privacy-preserving CS reservation and payment mechanism. As the literature is similar and is already presented in the above section, we just present the state-of-the-art literature on privacy-preserving energy trading.

### 1.3.1 Privacy-preserving energy trading

Several works have proposed privacy-preserving energy trading based on blockchain. Authors in (Samy, Yu, Zhang & Zhang, 2021) proposed a private blockchain-based secure energy transactions between energy sellers and buyers without a third-party. Authors in (Son, Im, Kwon, Jeon & Lee, 2020) proposed a P2P energy trading system on a blockchain where all bids are encrypted and peer matching is performed on the encrypted bids by a functional encryption-based smart contract. Authors in (Lu *et al.*, 2019) proposed a blockchain based privacy-preserving distributed transaction scheme for energy trading to maximize the protection of user privacy. Authors in (Radi *et al.*, 2019) proposed an anonymous payment system that cannot link individual owners to specific charging locations. Authors in (Baza & et al, 2021) proposed a privacy-preserving CS2V scheme to enable energy trading between CSs and EVs. They also proposed a privacy-preserving V2V energy trading scheme to enable EVs to charge other EVs. Authors in (Jiang, Zhang, Li, Yue & Zhou, 2020) proposed a privacy-preserving stealth transmission approach based on blockchain to ensure data privacy and break the linkage between consumers and providers in the energy trading process. Authors in (Long, Chen, Ren, Dou & Xiong, 2020) proposed a decentralized blockchain-enabled energy trading scheme which preserves the privacy of EV owners, by adopting the k-anonymity method in constructing a united request to hide the location information and creating a clocking area based on undirected graphs. Authors in (Hassan, Rehmani & Chen, 2021) proposed an energy trading mechanisms, which allows buyers and sellers to trade electricity without the risk of losing or compromising

their private data along with enhancing trust in the network. Authors in (Yang *et al.*, 2020) use the Ciphertext Policy Attribute-Based Encryption (CP-ABE) scheme to establish a blockchain based P2P energy trading approach with privacy preservation, which allows buyers and sellers, to manage and verify the transactions autonomously without needing third party intermediaries.

However, all the aforementioned work including the state-of-the-art work explained in previous section do not address end-to-end privacy-preserving slot reservations at CSs, i.e., they do not provide a mechanism to secure EVs' private information or the dissociation of physical identity at CS with blockchain addresses, which poses a serious threat to EVs' privacy and can expose complete charging history of a certain EV.

Solving this problem by just encrypting the data will not be useful as the data needs to be processed in the smart contract and those operations cannot be performed on encrypted data. Therefore, in Chapter 5, we extend the work in Chapter 4 and focus primarily on end-to-end privacy preservation for EVs at the time of charging slot reservation or payment for the charging services. More importantly, we enable EVs to reserve a charging slot at the selected CS privately, without sharing their private information with other entities in the network. Also, the proposed protocol protects the EV's privacy by dissociating its real identity from the blockchain address, thereby preserving the EV's privacy.

# CHAPTER 2

# BACKGROUND

We now provide a review of background knowledge, concepts, and techniques required for proper understanding of all aspects of this thesis.

## 2.1 Blockchain

Using the peer-to-peer (P2P) protocol, a network of machines called miners manage the chain of blocks that makes up a blockchain. According to a predefined consensus algorithm, each block contains a set of transactions committed by networking peers (Kosba, Miller, Shi, Wen & Papamanthou, 2016). At its inception, blockchain was used as a distributed cryptocurrency that allowed electronic payments to be made without the help of banks. Over the years, it has evolved to support the deployment of more general-purpose distributed applications. Vitalik Buterin introduced the concept of smart-contracts or decentralized autonomous organizations. (Wood *et al.*, 2014). In simple terms, a smart-contract is an autonomous computer program that runs on the blockchain network. Programs like this one are contracts whose terms can be pre-programmed with the ability to self-execute and self-enforce themselves without the intervention of trusted authorities (Christidis & Devetsikiotis, 2016).

Blockchain systems are typically composed of multiple nodes that do not fully trust each other. Each node maintains a set of global states, and each node performs transactions that may modify those states. Block chains are special data structures that store historical states and transactions. Each node in the system agrees on the order of transactions. Fig. 2.1 demonstrates how a transaction is verified and added to the blocks according to the blockchain protocol, in a P2P setting. As a result, blockchain is often referred to as a distributed ledger. The block contains the roothash (or Merkle root), which is the sum of all the transaction hashes (Christidis & Devetsikiotis, 2016). Miner needs to find the correct nonce for the block header in order to add the block to the chain.

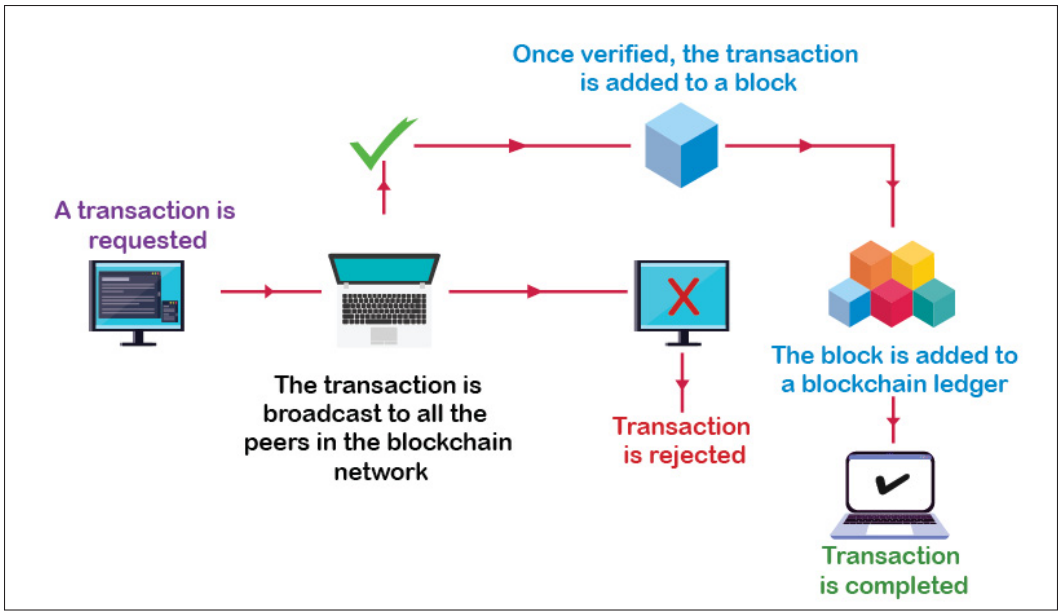



Figure 2.1 Illustration of how blockchain works

### 2.1.1 Features of blockchain

Blockchain offers several key features (Masood, Qureshi, Danish & Lestas, 2019) such as:

1. *Decentralized data management.* There is no one user in the system who owns the system more than any other, because each peer can add information to the ledger, which means that transactions can be made.
2. *Distributed ledger and transparency.* Every peer in the network has access to every transaction made with the help of a shared public list of transactions (the exchange of data).
3. *No central point of failure.* Due to the lack of a centralized storage system, there is no risk of losing data when a node is compromised.
4. *Data security, tamper-proof, anti-forgery and data integrity.* The blockchain architecture stores data in a way that is immutable and tamper-proof. As blockchains are decentralized, it is exceedingly difficult for attackers to modify ledgers.

### 2.1.2 Private vs public blockchains

From a high level, a blockchain system can be categorized into either public or private. With the former, any node can join and leave the system, so the blockchain is truly decentralized, similar to a peer-to-peer network. In the latter, the blockchain enforces strict membership. More specifically, the access control mechanism used to determine who may join the system is an authentication process, so that every node is known by the other nodes and their identities are known.

**Private blockchain:** Hyperledger (Aggarwal & Kumar, 2021) is among the popular private blockchains. Since node identities are known in the private settings, most blockchains adopt one of the protocols from the vast literature on distributed consensus. Zab (Junqueira, Reed & Serafini, 2011), Raft (Ongaro & Ousterhout, 2014), and PBFT (Castro, Liskov *et al.*, 1999) are popular consensus protocols that are widely used nowadays. Hyperledger directly uses PBFT consensus that has three-phases. In the pre-prepare phase, a leader broadcasts a value for other nodes to commit. The nodes then broadcast values they are going to commit in the prepare phase. In the final phase of the process, the committed value is confirmed when two thirds or more of the nodes agree with the previous value.

**Public blockchain:** Bitcoin is the most well known example of public blockchains. During a Bitcoin transaction, the states are digital coins (cryptocurrencies), and the coins are moved from one set of addresses to another. In this system, each node broadcasts a set of transactions it wants to execute. In the blockchain, special nodes, called miners, collect transactions into blocks, verify their validity, and start a consensus protocol to add them to the chain. The Bitcoin blockchain is based on a consensus mechanism known as proof-of-work (PoW), where only those miners who solved a computationally hard puzzle (finding the right nonce for the block header) are allowed to add new blocks to it. Additionally, Ethereum has established itself as one of the most well-known, public blockchains and is the second-largest cryptocurrency platform after bitcoin. It has its own cryptocurrency that is Ether.

```
pragma solidity 0.5.8;

contract SimpleStorage {
    event StorageSet(string _message);

    uint public storedData;

    function set(uint x) public {
        storedData = x;

        emit StorageSet("Data stored successfully!");
    }
}
```

Figure 2.2 An example of smart contract written in solidity

In this work, we are using a smart contract based public blockchain network in our architecture as it ensures pseudo-anonymity of the participating parties. The application users are communicating with the untrusted service providers on the blockchain network with trade related private information. Therefore, anonymity must be ensured to secure user's private information by using public blockchain network. Moreover, considering the anonymous payment feature of the public blockchain network, we also propose an anonymous payment mechanism. In this thesis, we have employed Ethereum blockchain to run the experiments. However, any public blockchain with smart contract functionality can be employed in the proposed system architectures.

### 2.1.3 Smart contracts

An smart contract is simply a program that executes whenever an electronic transaction is performed. In essence, it is a stored procedure that is invoked when a transaction occurs. Every node agrees to the inputs, outputs and states affected by the smart contract execution. All blockchains have built-in smart contracts that implement their transaction logics. A smart contract built into crypto-currencies verifies transactions by checking the signatures on transaction inputs. In the next step, it checks that the output addresses match the input addresses. Finally, it applies

changes to the states. One way to classify smart contracts is by its language. Fig. 2.2 shows an example of a smart contract written in Solidity that runs on Ethereum blockchain.

## 2.2 Machine learning and neural networks

Artificial Intelligence (AI)/Machine Learning (ML) allows computers to learn from experience. Algorithms for machine learning are designed to extract natural patterns and patterns from data by using computational methods and analyzing the process for extracting features. The two main types of machine learning techniques are supervised learning (SL) and unsupervised learning (UL) (Kumar & Selvakumar, 2011). The main difference between various types of machines is that supervised learning (SL) algorithms require input and output for them to learn a function. However, in unsupervised learning (UL), the algorithm uses only the input vector to learn about relationships between data (Kumar & Selvakumar, 2011).

### 2.2.1 Neural networks

Neural networks (NNs) are models of machine learning inspired by the structure and functioning of biological brains. A neural network consists of nodes, or neurons, which are simple computational units. An input is received at the edges of a neuron, multiplied by edge weights, and then subjected to a non-linear function called their activation function in order to achieve an output. Using a vector equation, a neuron's working can be modeled mathematically as in Eqn. 2.1, where x, w, b, $\odot$, f, y represent input vector, weight vector, neuron bias, element-wise multiplication, activation function, and neuron output respectively.

$$y(x) = f(w \odot x + b) \tag{2.1}$$

Typical activation functions include: logistic sigmoid ($\sigma$), tanh, and rectified linear units (ReLU) (Saied, Overill & Radzik, 2016). The functions are defined by equations 2.2, 2.3, and

2.4 respectively. For regression problems which require predicting continuous values, linear activation is used. Linear activation applies the identity function shown in equation 2.5.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.3}$$

$$ReLU(z) = max(0, z) \tag{2.4}$$

$$a(z) = z \tag{2.5}$$

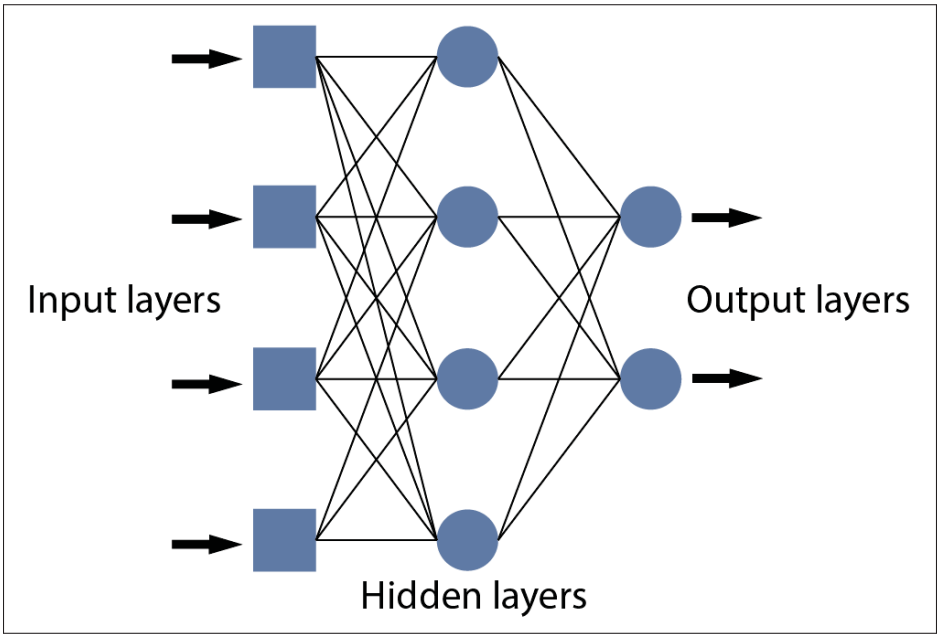



Figure 2.3 A simple neural network

The basic feed-forward network is shown in Fig. 2.3 and comprises of several neurons, also called units, organized in layers to form a network. A set of directed edges connects every neuron in each layer to every neuron in the previous layer, and a weight is assigned to each edge. The input layer is the first layer that receives the input. NN's output is produced by the last layer, called the output layer. We refer to the remaining layers collectively as hidden layers. Layer structure implies a hierarchy because information flows from the input layer to the output layer.

A layer's input and output are also called the bottom and top layers, respectively. A neuron's output is calculated as it moves upwards from the bottom layer to the top layer.

### 2.2.2 Recurrent neural networks

In NNs and, in fact, many other machine learning models, sample independence is one of the major assumptions. Data that is sequential, however, does not conform to this assumption. All of these elements exhibit interdependence across time, whether they are speech, language, time series, video, etc. Each sample of data is treated individually by NNs, so they lose the opportunity to exploit sequence information.

RNNs are a special type of NN that process sequential data. RNNs have a state vector (in the hidden units) that stores the memories of all previous elements of a sequence. The most simple RNN is shown in Fig. 2.3. As can be seen, the hidden neurons in an RNN are connected via a feedback connection. They maintain a hidden state vector as a memory for past information as they process the input sequence one element at a time. By learning to selectively retain relevant information, they can capture dependencies across multiple time steps. When making future predictions, they can use both current input and past information. In RNN training, the model is unfolded and a copy is created for each time step. RNNs that have been unfolded can be treated as NNs with multiple layers and trained the same way as back-propagation. BPTT, or back-propagation through time, is the method used to train RNNs (Larson, 2016).

## 2.3 Ring signatures

The ring signatures concept was first introduced in (Rivest, Shamir & Tauman, 2001) by Rivest, Shamir, and Tauman. With the help of a ring signature, a signer from a set of possible members convinces the verifier that the signer belongs to the group without disclosing his identity. A verifier can verify the correctness of the ring signature, i.e., the signature comes from some member of the group, but it cannot tell exactly who the signer is thus, the signer's anonymity is ensured.

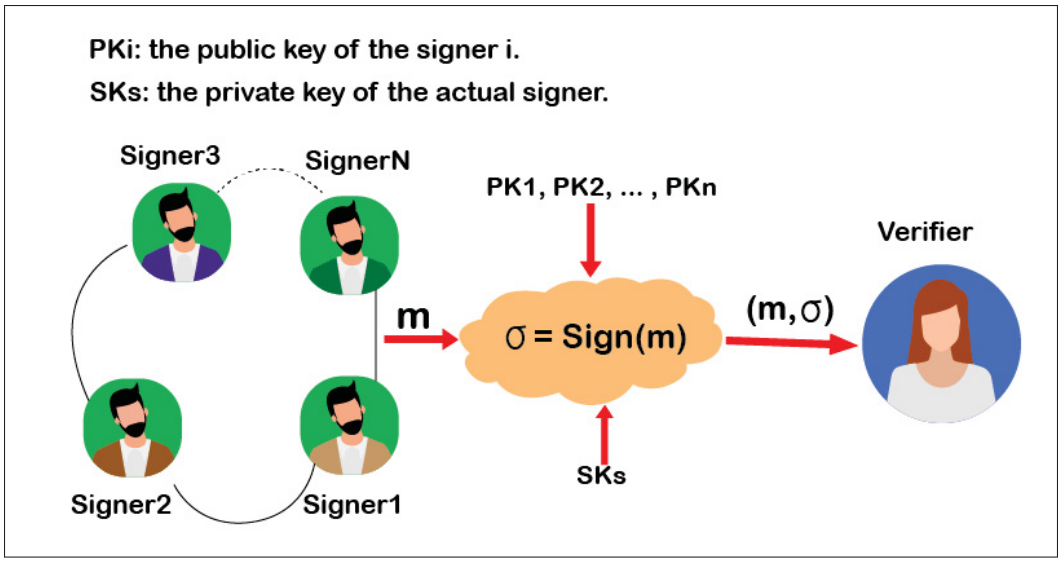



Figure 2.4 Illustration of how ring signatures work

Ring signatures use public key cryptography, i.e., a public key and a corresponding private key to securely transmit messages. The public key is used to encrypt a message, while the corresponding private key is used to decrypt a message. These public and private key pairs are generated using ECC, and the public key is shared among all the parties. A ring signature is defined with two procedures:

- *Ring Sign*(Message, $N$ user's public keys, signer's private key) - Generate a ring signature with message $M$, public keys of ring members, and signer's private key.
- *Ring Verify*($M$, $\sigma$) - Verify ring signature with message $M$ and ring signature $\sigma$ and return 1 if a signature is valid.

Fig. 2.4 shows how signatures are generated in the form of ring and sent to the verifier. We argue in favor of ring signatures to ensure EVs anonymity because ring signatures hide EVs' identity from the RSU as well as the CS. Moreover, ring signature is implemented off-chain between EVs and RSUs, therefore, EVs do not need to communicate with CSs directly, thus ensuring privacy.

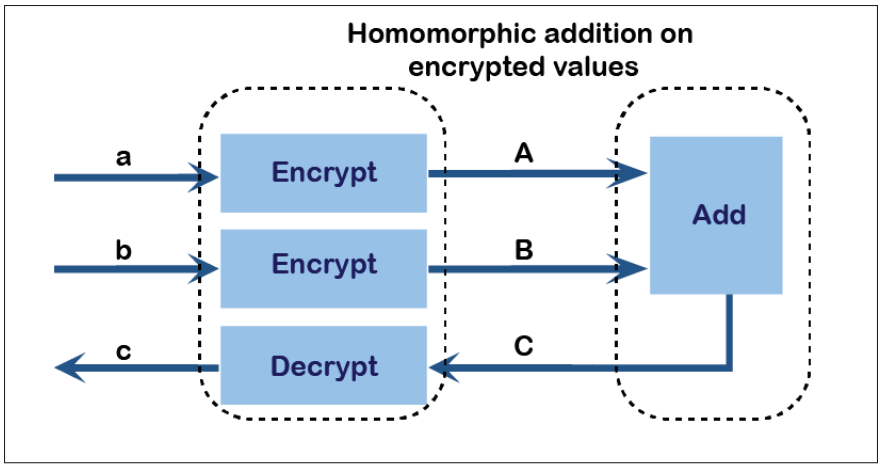



Figure 2.5 Additive homomorphic encryption example

## 2.4 Homomorphic encryption

SMC is a cryptographic problem in which multiple parties jointly compute a value based on individually held private data, without sharing the data. A more general representation of SMC is: for different number of parties $P_1$,...,$P_n$ with an initial inputs $m_1$,...,$m_n$, SMC securely computes a function $f(x_1, ..., x_n) = k_1$,...,$k_n$, where $k_1$,...,$k_n$ is the output of the function, which is received by each party, while keeping each party's input private.

In this work, we use homomorphic encryption based on the Paillier cryptosystem (Paillier, 1999), a standard approach to SMC, which enables direct arithmetic operations on encrypted values. The encrypted information of multiple parties can be added up without having to know the actual value of each party. One simple illustrative example would be: Let n = $pq$, and $p$ and $q$ be distinctive large prime numbers, where $(p, q)$ is the private key and $n$ is the public key. The encryption properties of Paillier-based homomorphic encryption makes it possible to add two encrypted messages, i.e., given two encrypted messages m1 and m2, and a constant number n, one can calculate $Enc(m1 + m2) = Enc(m1).Enc(m2)$ and $Enc(m1)^{m2} = Enc(m1m2)$. As shown in Fig. 2.5, two numbers, 0 and 1 are encrypted before, and the addition functionality is implemented on the encrypted numbers. Thus, the homomorphic encryption will output the correct answer given the encrypted inputs.

We argue in favor of homomorphic encryption based SMC to CS charging slot information because our main goal is to add encrypted information in a decentralized way. With the help of homomorphic encryption, EVs encrypt and share information with each other to calculate the available and occupied charging slots without the need of trusted third party.

## 2.5 Trusted execution environment

We propose a conflict resolution mechanism based on TEE that protects the confidentiality and integrity of computations, and can issue proofs, known as attestations, of computation correctness. Intel SGX is a specific TEE technology, which is used in our proposed framework. Intel SGX provides a CPU-based implementation of TEEs—known as enclaves in SGX. A host can instantiate an enclave which is isolated from the host itself. The enclave provides a protected address space to the code running inside a TEE. When data from a TEE moves off the processor to memory, it is transparently encrypted with keys only available to the processor. Thus the operating system, hypervisor, and other users cannot access the enclave's memory. The SGX memory encryption engine also guarantees data integrity. Moreover, Intel SGX supports attested execution, i.e., it is able to prove the correct execution of a program, by issuing a remote attestation, a digital signature, using a private key known only to the hardware, over the program and an execution output. Assuming trust in the hardware, and Intel, which authenticates attestation keys, it is infeasible for any entity other than an SGX platform to generate any attestation, i.e., attestations are existentially unforgeable.

We argue in favor of TEE based conflict resolution mechanism because smart contract directly takes input from the smart meters of EVs and CSs and process it without the need of a trusted third party. TEE ensures that the correct input is coming from the smart meters of EVs and CSs with the help of remote attestation.

# CHAPTER 3

# BLOCKAIM: A NEURAL NETWORK-BASED INTELLIGENT MIDDLEWARE FOR LARGE-SCALE IOT DATA PLACEMENT DECISIONS

Syed Muhammad Danish[1] , Kaiwen Zhang[1] , Hans-Arno Jacobsen[2]

[1] Département de génie logiciel et des TI, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Electrical & Computer Engineering, University of Toronto

Paper published in *IEEE Transactions on Mobile Computing*, April 2021

The content of this chapter concerns a blockchain-based middleware for large-scale IoT data storage selection. Section 3.1 presents various motivating use cases for this work. Section 3.2 explains the proposed middleware based IoT fog architecture. Section 3.3 explains the blockchain-based data placement protocol and theoretical problem formulation of the proposed middleware design. Section 3.4 explains the proposed neural network based maintenance strategy and reconfiguration. Section 3.5 presents the theoretical formulation of the aggregation rate feedback. Section 3.6 presents the middleware design evaluation. Finally, a conclusion is offered in Section 3.7.

## 3.1 Introduction

Cloud technology has emerged as a promising and scalable solution to store and manage large-scale IoT data (Khaled, Helal, Lindquist & Lee, 2018; Sun, Yin, Sun, Tian & Du, 2020). Normally, the raw heterogeneous data is generated by the IoT devices, which needs to be preprocessed first to remove the redundant information and extract meaningful insights before storing. A fog/edge server (Debe & et al., 2019) performs the filtering and aggregation of the raw IoT data and operates near to the IoT end devices. The time-critical IoT data is processed by the fog nodes and the archived data is sent to the cloud premises for retrieval and storage in the future, as shown in Fig. 3.1 (Mollah & et al, 2017).

Unfortunately, the current centralized cloud storage technologies suffer from significant privacy, security and trust issues (Li *et al.*, 2018). As IoT devices create huge volumes of data, much of it is sensitive data belonging to individuals and companies can easily be leaked from the cloud servers (Li *et al.*, 2018). In addition, data errors can be concealed by the cloud storage providers intentionally, which directly affects the integrity of the large-scale IoT data (Wang & et al, 2009). Thus, it is necessary to assure the IoT application owners that the data has been maintained correctly throughout the period of data storage (Kumar & Saxena, 2011). Moreover, while dealing with untrusted storage technologies, accountability, traceability and auditability of the data storage decisions, data failures, data provenance and access records are important to ensure confidence and trust (Ko, Lee & Pearson, 2011). Currently, a Third-Party Auditor (TPA) performs the data integrity verification and auditing on the behalf of users and are assumed to be honest. However, in practice, TPAs are not reliable and can be compromised (Zhang, Xu, Lin & Shen, 2019), which poses serious threat to the user's data privacy (Bai & Hao, 2019).

In recent times, decentralized storage solutions, based on a combination of the blockchain technology (Zhang & et al, 2018; Jiang, Fang & Wang, 2018) and peer-to-peer (P2P) networking (R & et al, 2019) have gained attraction as an alternative to cloud storage technologies. These decentralized storage technologies aim to give more control to the clients over their data. Examples of emerging decentralized storage solutions include Storj, Filecoin, Safecoin (L & et al, 2015), and Sia (Vorick & Champine, 2014) etc. Blockchains eliminate the need of trusted third party as the ledger is maintained by all the nodes in the network instead of a centralized server. Moreover, it offers transparency, immutability, availability and verifiability in an untrusted environment (Chen & et al., 2019) as the information stored on the network cannot be changed and is visible to all the nodes in the network. Moreover, the data cannot be modified or deleted once it is stored in the blockchain network (Danish *et al.*, 2020a). Therefore, in the light of the above benefits, blockchain solution can be leveraged to provide accountability, auditability and data integrity to the IoT application without the need of trusted third parties or TPAs (Masood & et al., 2019).

Selecting the right data storage technology for the given requirements is a key design decision. IoT application developers now face a challenge in selecting the best storage candidate for their service requirements, considering the large variety of cloud and decentralized storage options available and their underlying heterogeneous storage systems, as well as their different promised Service Level Agreement (SLA) guarantees (Rafique & et al, 2019). Traditionally, IoT applications infrastructure relies only on a single cloud storage technology, which limits the possible service requirements of IoT applications. In addition, relying on a single cloud storage technology increases the risk of technology or vendor lock-in and places limits on the application and SLA requirements (Rafique & et al, 2017). Multi-cloud heterogeneous architecture (Wu & et al, 2013; AlZain & et al, 2011; Dobre & et al, 2014; Cachin *et al.*, 2010; Bowers & et al, 2009) has recently been proposed, but with limited impact due to the lack of differentiation between competing cloud solutions. In contrast, decentralized storage solutions' architecture differs significantly from cloud technologies and possess distinct characteristics in terms of security, privacy, performance, price and availability (Benisi, Aminian & Javadi, 2020). Relying on a single storage technology may lead to sub-optimal data placement decisions in highly dynamic environments (Rafique & et al, 2017). Hence, there is a need to take these parameters into account while selecting the storage technology for IoT applications.

In this paper, we study the problem of data placement decision, which is the selection of an effective storage technology to fulfill the service requirements and parameters of a particular IoT application by jointly considering cloud and decentralized storage. Important factors include the volume of data generated, privacy requirements, and price considerations.

Existing works have employed the blockchain-based solutions to address the large-scale IoT data storage problem (Shafagh & et al, 2017; Dai & et al., 2019; Li *et al.*, 2018; Dorri & et al, 2017; Karlsson & et al, 2018). However, these works focus on the IoT data management aspect and consider a single off-chain solution to store the IoT data. Therefore, the problem of considering multiple storage solutions to select the best-suited storage technology for large-scale IoT data based on the IoT application's service requirements simultaneously with the auditable data provenance has been largely ignored in blockchain-based solutions. Furthermore, the

current middleware storage selection solutions only consider cloud storage technology. Also, the previous works lack the theoretical formulation of IoT data placement problem and ignore the captivating properties of the middleware, e.g., accountability, auditability, traceability and decision verifiability for both the storage technologies as well as IoT applications. Finally, the current middleware solutions constantly monitor the storage technologies to retrieve the parameters and lack adaptability, which leads to high computation and communication cost.

In our previous work (Danish & et al., 2020), we proposed an extended fog architecture for IoT application with a blockchain network. We theoretically formulated a data storage selection problem as a decision optimization problem that finds the storage solution based on the captured cost functions of storage technologies and service requirements of IoT applications, and proposed two polynomial-time heuristic algorithms to approximate a solution to the formulated problem. Compared to our previous work (Danish & et al., 2020), BlockAIM ensures the data placement decision correctness verifiability by IoT applications as well as storage technology. Data placement decision verification is important from storage technology as well as IoT application's perspective, i.e., to make sure that the middleware has used the correct parameters to select a storage technology. Moreover, it leverages neural network to learn and predict the dynamic behaviour of the IoT applications and storage technologies due to its effectiveness against conventional regression and statistical models, and its ability to deal with large volumes of IoT data (Al-Hawawreh, Moustafa, Garg & Hossain, 2020). Neural network enables the middleware design to adapt itself intelligently to make the data placement decisions for the IoT applications. First, the middleware design predicts IoT service requirements and the next re-evaluation period with the help of a neural network-based LSTM network. Second, the middleware decides whether to re-evaluate the data placement decision, based on the predicted parameters. Finally, the proposed middleware design intelligently tunes the aggregation rate for the raw IoT data to optimize the data quality and precision defined by the IoT application owner.

Our main contributions in this work are:

- We propose BlockAIM: a neural network-based middleware IoT fog architecture for data placement decisions of IoT applications. We model the optimal data placement problem

as the NP-hard Uncapacitated Facility Location problem (UFL) and provide two heuristics: dynamic programming-based (DP) and greedy-style heuristic (GS).

- We propose a verifiable blockchain-based storage selection protocol, which enables the IoT applications as well as the storage technologies to verify the correctness of the data placement decision made by the middleware design without the need of any trusted third party. The proposed smart contract protocol ensures accountability, traceability and data integrity for IoT applications as well as storage technologies.
- We propose an intelligent maintenance strategy combining neural networks and Long Short-Term Memory (LSTM), which takes into account and learns the dynamically evolving features of the IoT applications service requirements to enable maintenance reconfiguration and optimizes the computational complexity along with the blockchain storage and transaction overhead in the middleware design.
- We propose a feedback mechanism to control the data aggregation rate, and tune data quality and precision. We model the aggregation rate feedback problem as a linear optimization problem, which optimizes the price of the data storage, while maximizing the data quality and precision of the specified IoT application.
- We perform extensive experiments to evaluate the proposed middleware design along with the proposed heuristics' performance. Our results show that our middleware effectively reduces the price of IoT data storage while maintaining service requirements.

## 3.2 Proposed middleware architecture

In this section, we propose a neural network-based adaptive middleware design, which jointly considers decentralized storage technologies along with cloud storage. Compared to the traditional IoT fog architecture, the proposed architecture includes an adaptive middleware module to intelligently select the storage technology for IoT applications and to learn the dynamic behaviour of parameters for future data placement decision predictions. The proposed design is shown in Figure 3.2.

**Network Entities Definition:**

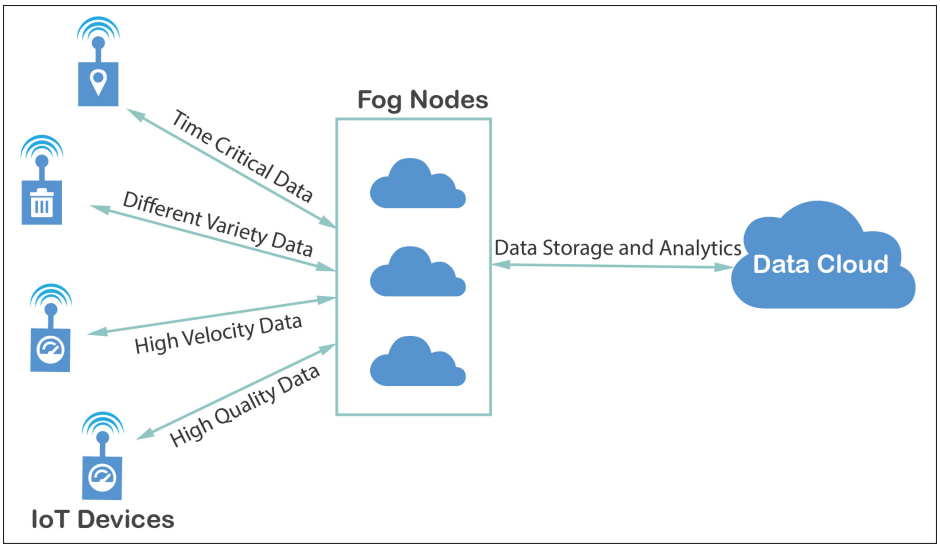



Figure 3.1 Traditional cloud architecture for iot applications

1. *Fog Nodes:* Fog nodes operate near IoT devices, at the network edge, to extract time-sensitive information and perform real-time computation on the raw IoT data. In this work, we assume the fog nodes to be honest, and the communication between the IoT devices and the fog node is assumed to be secure because it resides in the local network of IoT applications (Vaquero & Rodero-Merino, 2014). Moreover, fog nodes also operate in hierarchical architecture with multiple levels to process time-sensitive data. The fog node has three main responsibilities: First, it performs aggregation and filtering on the IoT data. Second, it writes the data hash and application ID on the blockchain network to provide data integrity. Third, it keeps track of up-to-date service requirements.
2. *Blockchain Network:* Fog nodes publish the IoT application's service requirements, while the storage technologies publish their service parameters on the blockchain network. BlockAIM utilizes these parameters for storage selection and publishes the data placement decision on the blockchain network, thus ensuring trust and verifiability for both the IoT applications and the storage technologies. In order to achieve these goals, we employ public blockchain smart contracts functionality. It should be noted that the blockchain module in our proposed architecture in Figure 3.2 is a separate entity and working independently from the blockchain network of decentralized storage technologies.

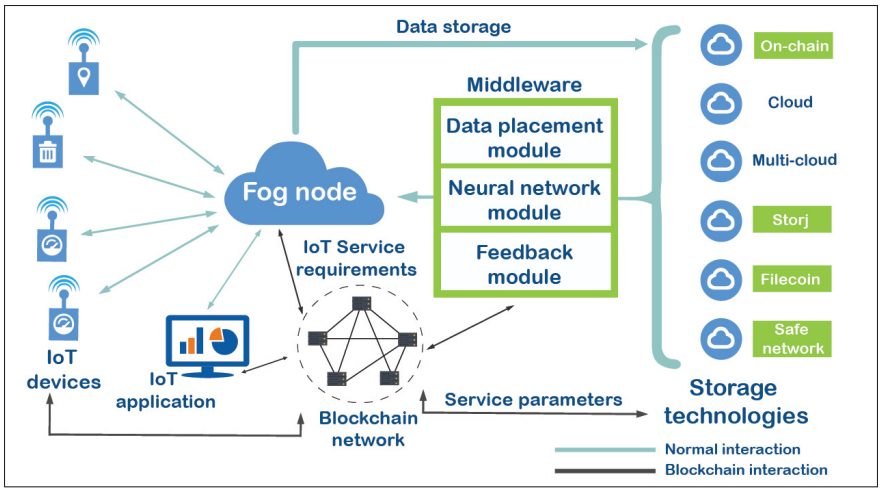



Figure 3.2 Middleware architecture (Danish, 2019)

3. *Storage Technologies:* Each storage technology handles the read and write requests in accordance to the decision made by BlockAIM. A representative list of available storage technologies is shown in the Figure 3.2.
4. *Adaptive Middleware:* The middleware module of our proposed architecture is a piece of software, running on the fog nodes and is directly connected to the blockchain network and the storage technologies. The middleware has three main responsibilities: First, it makes the blockchain-based data placement decisions based on the IoT service requirements and service parameters of storage technologies. Second, it learns the dynamic behaviour of IoT applications as well as storage technologies with the help of neural network algorithms and adapts itself to intelligently schedule the data placement decisions. Third, it provides the feedback to IoT applications to control data quality and precision. These responsibilities are described in detail in Sections 3.3-3.5.

## 3.3 Blockchain-based storage selection protocol

In this section, we propose a blockchain-based data placement protocol for IoT applications, which selects the storage technology for the IoT applications, based on their service requirements.

### 3.3.1 Overview of the storage selection protocol

We first provide an overview of the entire storage selection protocol. Details about important steps, i.e., the decision optimization problem and the re-evaluation interval will be given in the following sections. A blockchain network is employed to store the IoT applications service requirements, service parameters of the storage technologies and the data placement decisions by interacting with the smart contract. Thus, the blockchain network is used as a storage and decision verification layer to provide traceability, auditability and accountability to the IoT applications as well as storage technologies. Data placement decision verification is important from storage technology as well as IoT application's perspective, i.e., to make sure that the middleware has used the correct parameters to select a storage technology. More details about the blockchain-based storage selection protocol are given below:

**Storage technology registration:** First, all of the storage technologies including decentralized storage technologies and public cloud infrastructures, will register themselves to the middleware by providing a valid blockchain address and ID, namely,

$$ST_m \rightarrow MW : request = \{ID||address_{bc}\}$$

where ST represents the storage technology, while middleware is represented by MW. Once the request has been received, the middleware will register the new storage technology in the list of available storage solutions.

**Blockchain-based protocol:**

- *Step 1:* All the storage technologies will periodically write their service parameters information on the blockchain network at time $t$. These service parameters include the storage, retrieval and bandwidth price, latency, availability and privacy, namely,

  $$ST_m \rightarrow Blockchain : input = \{price||latency||availability||bandwidth||privacy\}.$$

  The information update at the time $t + 1$ will replace the service parameters information at the time $t$. Therefore, the middleware will always utilize up-to-date service parameters

for data placement decisions. $Events$ functionality in the smart contracts can be utilized to notify middleware about the parameters update. It should be noted that the new service parameters related to the storage technologies can also be identified and incorporated in the current problem.

- *Step 2:* Similarly, the fog nodes periodically update the service requirements of multiple IoT applications on the blockchain network at time $t$. These service requirements include the data size, data retrieval frequency price and performance requirements, and are specific to the individual IoT application, namely,

$$Fog \rightarrow Blockchain : input = \{datasize||price||retrievalfreq||performance\}.$$

Similar to *step 1*, the service requirements updation at time $t+1$ will replace the requirements at time $t$.

- *Step 3:* In order to select the storage technology for a given IoT application, middleware will take the up-to-date parameters of IoT applications as well as storage technologies and solve a decision optimization problem. We formally formulate the storage selection optimization problem in the following Section 3.3.2. Once the middleware has made the data placement decision, it will notify the fog nodes and storage technology of the decision as well as write on the blockchain network as:

$$MW \rightarrow Fog, ST : notify = \{ID_{IoT}||ST\}$$

$$MW \rightarrow Blockchain : input = \{ID_{IoT}||ST||timestamp\},$$

where $ID_{IoT}$ represents IoT application ID. It should be noted that the solution to the optimization problem will be approximated off-chain by the middleware as the data placement decisions need to be made in real time. A significant delay as well as high storage and transaction cost will be incurred if this problem is solved in the smart contract on public blockchain.

- *Step 4:* Once the data placement decision has been taken, the IoT applications and the storage technologies can verify the correctness of the decision by retrieving the corresponding parameters from the blockchain network as follows:

$$Blockchain \rightarrow Fog : retrieve_{ST} = \{price||latency||availability||bandwidth||privacy\}.$$

$$Blockchain \rightarrow ST : retrieve_{IoT} = \{datasize||price||performance\}.$$

It must be noted that the decisions of the middleware design cannot be verified in the real time, and can only be verified in future. It is possible that the middleware makes the wrong decision in the real-time, and there is no way to verify it in real-time. However, this decision can be audited in the future using the storage and application parameters stored in the smart contract.

- *Step 5:* Once the storage technology has been selected at time $t$, based on the service requirements of IoT applications and service parameters of storage technologies, BlockAIM will predict the time $t + 1$, i.e., when to re-make the data placement decision. Details of these predictions are given in Section 3.4.

### 3.3.2 Storage selection problem formulation

**System Model:** We consider an IoT network consisting of $\boldsymbol{D}$ IoT applications, which continuously generate large-scale data with different data types $I$ independently. The generated data is first pre-processed at the fog node and the aggregated data is then stored in different storage technologies. We consider $\boldsymbol{J}$ storage technologies to store the large scale IoT data, including public clouds, decentralized storage technologies and multi-cloud as shown in Figure 3.2. After pre-processing of raw IoT data, fog node produces $F_{d,i}$ bytes of data with data type $i \in I$.

**Decision Variable:** We define a binary decision variable $y_j$, indicating whether the storage solution $j$ is selected ($y_j = 1$) or not ($y_j = 0$) for IoT application $d \in D$ with data type $i \in I$. Let $\overline{y} = (y_j) \forall j \in J$ represents the set of all the storage variables and is given by:

$$\mathcal{Y} = \{\, \overline{y} \mid \textstyle\sum_{j \in J} y_j \leq J \text{ and } y_j \in \{0, 1\}, \forall j \in J \,\}$$

**Overall Cost:** We define the total cost $C_t$, which consists of five components: bandwidth cost, availability cost, retrieval and storage cost, latency cost, and privacy cost.

(1) The ***bandwidth cost*** denotes the price for downloading and uploading the data for a specific storage technology. If $P_j$ is the bandwidth price of $j^{th}$ storage technology and $\sum_{i \in I}(F_{d,i})$ is the total amount of data generated by an IoT application, we can represent the overall bandwidth cost as Equation (3.1):

$$C_{BW}(y) \sum_{j \in J} \sum_{i \in I} y_j P_j F_{d,i} x_i, \forall i \in I, j \in J, \forall j \in J \exists ! i : x_i \tag{3.1}$$

where $x_i \in \{0, 1\}$ represents whether the selected data type $i$ is considered for the storage technology $j$, while the condition $\forall j \in J \exists ! i : x_i$ states that only one storage solution must be selected for one data type $I$ of an IoT application.

(2) The ***retrieval and storage cost*** represent the monetary cost for storage and retrieval of the IoT data. Let $\sum_{i \in I}(F_{d,i})$ and $\sum_{i \in I}(R_{d,i})$ be the total amount of data that needs to be stored and retrieved respectively. Let $Hj$ and $Gj$ represent the per byte retrieval and storage cost for $j^{th}$ storage technology respectively. Equation (3.2) represents the overall storage cost as:

$$C_{CS}(y) \sum_{j \in J} \sum_{i \in I} y_j (GjF_{d,i} + HjR_{d,i}) x_i, \quad \forall i \in I, j \in J, \forall j \in J \exists ! i : x_i \tag{3.2}$$

(3) The ***latency cost*** denotes the latency to store or retrieve the data from storage technology. $T_b$ and $l_j$ represent the block time and link latency respectively. We can denote the incurred latency cost for uploading the overall data $\sum_{i \in I}(F_{d,i})$ as Equation (3.3):

$$C_{LT}(\overline{y}) \sum_{j \in J} \sum_{i \in I} W_L \left(\frac{F_{d,i}}{BW_j} y_j x_i + l_j + T_b\right), \quad \forall i \in I, j \in J, \quad \forall j \in J \exists ! i : x_i \tag{3.3}$$

where $BW_j$ is the bandwidth provided by $j^{th}$ storage technology. Since, the latency cost is not represented in monetary unit, we define a parameter $W_L$, which represents the monetary cost weight associated to $C_{LT}$.

(4) The ***availability cost*** represents the probability for the data to be online during the retrieval requests. Equation (3.4) represents the overall availability cost as:

$$C_{AV}(\overline{y}) \sum_{j \in J} W_A y_j \alpha_j, \ \ \forall j \in J \tag{3.4}$$

where $\alpha_j$ represents the availability of $j^{th}$ storage technology, i.e., $0 \leq \alpha_j \leq 1$. Since, the availability cost is not represented in monetary unit, we define a parameter $W_A$, which represents the monetary cost weight associated to $C_{AV}$.

(5) The ***privacy cost*** denotes whether the user privacy is offered by the required storage technology or not. Let the privacy parameter associated with the $j^{th}$ storage technology is $\beta_j \in \{0, 1\}$, i.e., $\beta_j = 1$ if the storage technology offers privacy, and 0 otherwise. Equation (3.5) represents the overall privacy cost as:

$$C_{PR}(\overline{y}) \sum_{j \in J} W_P y_j \beta_j, \ \ \forall j \in J \tag{3.5}$$

Since, the privacy cost is not represented in monetary unit, we define a parameter $W_P$, which represents the monetary cost weight associated to $C_{PR}$.

**Optimization Problem:** The objective function that needs to be maximized consists of privacy and availability cost, i.e., $maximize(C_{AV} + C_{PR})$, while latency cost, retrieval and storage cost and bandwidth cost need to be minimized, i.e., $minimize(C_{LT} + C_{CS} + C_{BW})$. we can denote the overall cost as Equation (3.6):

$$\mathbb{C}(\overline{y}) = C_{BW}(\overline{y}) + C_{CS}(\overline{y}) + C_{LT}(\overline{y}) - C_{AV}(\overline{y}) - C_{PR}(\overline{y}) \tag{3.6}$$

Thus, the optimization problem is formulated as Equation (3.7a):

$$\min \quad \mathbb{C} \tag{3.7a}$$

$$\text{subject to} \quad \overline{y} \in \mathcal{Y}. \tag{3.7b}$$

$$\sum_{j \in J} P_j \sum_{i \in I} F_{d,i} \leq \mathcal{B}, \forall i \in I, \forall j \in J \tag{3.7c}$$

$$\sum_{j \in J} (Gj + Hj) \sum_{i \in I} F_{d,i} \leq \mathcal{M}, \forall i \in I, \forall j \in J \tag{3.7d}$$

$$l_j \leq \mathcal{L}, \forall j \in J \tag{3.7e}$$

$$\alpha_j \geq \mathcal{A}, \forall j \in J, \tag{3.7f}$$

$$\beta_j \geq \mathcal{B}, \forall j \in J, \tag{3.7g}$$

where, $\mathcal{B}$, $\mathcal{A}$ and $\mathcal{L}$ represent the maximum defined value for price, availability and latency, respectively. Constraint (3.7c) states that $C_{BW}$ should not exceed $\mathcal{B}$, i.e., the bandwidth budget. Moreover, (3.7d) states that $C_{CS}$ should not exceed $\mathcal{M}$, i.e., the monetary budget. Constraints (3.7e), (3.7f) and (3.7g) ensure that $C_{AV}$, $C_{LT}$ and $C_{PR}$ should be less than the thresholds defined by the owner, i.e., $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{B}$ respectively.

The data centers selection and placement problems have been shown to be related to the Facility Location Problem (FLP) (B & et al, 2008; Z & et al, 2019). In a basic FLP problem, clients demands and set of facilities are used as an input, to find and open the facilities with minimum overall cost, based on different demands of clients. This is proven to be a well-known NP-hard problem (Guha & Khuller, 1999). Our formulated optimization problem in Eqn. (3.7a) is similar to the FLP variant problem, i.e., metric uncapacitated facility location problem (UFL) problem, in which the IoT applications' service requirements correspond to the clients demands and the storage technologies represent the available facilities.

**Theorem 1.** The optimization problem formulated in Equation. (3.7a) is NP-hard.

*Proof:* A proof can be found in our previous work (Danish & et al., 2020). Motivated by (G & et al, 2019), we construct a polynomial-time reduction to our cost minimization optimization problem (3.7a) from the UFL problem, a well known NP-hard problem (Guha & Khuller, 1999):

$$\text{min} \quad \sum_{i=1}^{n} f_i z_i + \sum_{i=1}^{m} \sum_{i=1}^{n} c_{i,j} u_{i,j} \tag{3.8a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} u_{i,j} = 1, \forall i \in I \tag{3.8b}$$

$$u_{i,j} \leq z_i, \forall i \in I, \forall j \in J \tag{3.8c}$$

$$z_i, u_{i,j} \in \{0, 1\}, \forall i \in I, \forall j \in J \tag{3.8d}$$

Our optimization problem consist of variable and fixed elements, similar to UFL problem. The fixed cost component are comprised of $C_{PR}$, $C_{AV}$ and can be represented as $C_{fixed} = (\sum_{j \in J} W_A \alpha_j + W_P \beta_j) y_j$ , while variable cost is represented as $C_{CS}, C_{BW}, C_{LT}$ and can be written as $C_{var} = \sum_{j \in J} \sum_{i \in I} (P_j F_{d,i} x_i + (Gj + Hj) F_{d,i} x_i + W_L l_j F_{d,i} x_i) y_j$. However, our optimization problem maximizes the fixed cost, compared to the minimization in UFL problem. Thus, we introduce two new parameters, i.e., $\hat{C_{AV}}, \hat{C_{PR}}$ to modify our optimization problem in Equation. (3.7a). $\hat{C_{PR}}$ represents the cost of $no - privacy$ as compared to Equation (3.5) and represented as $\hat{C_{PR}}(\overline{y})$ $\sum_{j \in J} W_P y_j \hat{\beta_j}$, $\forall j \in J$, where $\hat{\beta_j}$ represents no-privacy value, i.e., $\hat{\beta_j}$ = 1 - $\beta_j$. Similarly, $\hat{C_{AV}}$ states the $non - availability$ cost and represented as $\hat{C_{AV}}(\overline{y})$ $\sum_{j \in J} W_A y_j \hat{\alpha_j}$, $\forall j \in J$, where $\hat{\alpha_j}$ represents non-availability, i.e., $\hat{\alpha_j}$ = 1 - $\alpha_j$. Unlike fixed costs, $\hat{C_{AV}}, \hat{C_{PR}}$ need to be minimized. Thus, the optimization problem can be re-written as as Equation (3.9a):

$$\min \quad \sum_{j \in J} (\hat{C_{AV}} + \hat{C_{PR}}) + \sum_{j \in J} \sum_{i \in I} (C_{BW} + C_{CS} + C_{LT}) \tag{3.9a}$$

$$\text{subject to} \quad \overline{y} \in \mathcal{Y}. \tag{3.9b}$$

$$\sum_{j \in J} P_j \sum_{i \in I} F_{d,i} \leq \mathcal{B}, \forall i \in I, \forall j \in J \tag{3.9c}$$

$$\sum_{j \in J} (Gj + Hj) \sum_{i \in I} F_{d,i} \leq \mathcal{M}, \forall i \in I, \forall j \in J \tag{3.9d}$$

$$l_j \leq \mathcal{L}, \forall j \in J \tag{3.9e}$$

$$\alpha_j \leq \mathcal{A}, \forall j \in J, \tag{3.9f}$$

$$\beta_j \leq \mathcal{B}, \forall j \in J, \tag{3.9g}$$

Given an instance $I$ of the UFL problem, where $I = (m, n, c_{ij}, f_i)$, we map it to a problem instance of our cost minimization optimization problem (3.9a) with $\hat{I} = (|J| = n, |I| = m, C_{var} = c_{ij}, C_{fixed} = f_i, y_i = u_{i,j})$. It can be seen that the mapping of problem instance $I$ to $\hat{I}$ can easily be done in polynomial time (G & et al, 2019). Thus, if some algorithm solves the cost minimization optimization problem (3.7a) with problem instance $\hat{I}$, it also solves the related UFL problem with problem instance $I$ as well. Thus, we can consider the UFL problem as a special case of our cost minimization optimization problem. As the UFL problem is NP-hard, our cost minimization optimization problem must be NP-hard as well.

In this work, we considered the service parameters of storage technologies taken from state-of-the-art research work. The QoS service parameters list considered in this work is not complete, and in future, there is a possibility for the identification of more service parameters of the storage technologies. These parameters can easily be integrated with the optimization problem as a subject to conditions. Moreover, the proposed problem formulation is not limited to the decentralized storage technologies. One can simply add or remove the decentralized storage technologies, or introduce a new type of storage technology with associated parameters in the proposed model. However, increasing the number of storage technologies and introduction of

more service parameters in the optimization problem will increase the complexity of the problem. This could lead to the optimization of proposed algorithms or to introduce new algorithms to solve the problem in polynomial-time.

Algorithm 3.1 DP-heuristic

```
1  Initialize ȳ, T(y_j, C_j(ȳ))
2  ȳ ← 0
3  T(y_j, C_j(ȳ)) ← 0
4  for x ← 1 to i do
5  |   for y ← 1 to j do
6  |   |   Find all the parameters costs associated with y_j if parameters cost < Eqn
   |   |    (3.7c) − (3.7g) then
7  |   |   |   Assign y_j storage technology to x_i IoT application
8  |   |   |   Store the parameters cost value in T(y_j, C_j(ȳ))
9  |   |   |   break
10 |   |   end if
11 |   end for
12 end for
13 Find min_{∀ȳ∈𝒴} T(y_j, C_j(ȳ)) s.t. Eqn (3.7c) − (3.7g)
14 if no storage solution found then
15 |   Go to step 5
16 end if
```

### 3.3.3 Proposed solution

To approximate a solution to the NP-hard optimization problem formulated in Section 3.3.2, in this section, we propose two polynomial-time heuristic algorithms: dynamic programming-based (DP) heuristic and greedy style-based (GS) heuristic.

**DP-based heuristic:** The DP-based algorithm computes the corresponding parameters cost (Equation. (3.1) - (3.5)) for different storage technologies and use first fit strategy to select $\overline{y}$. A 2-D table is constructed $T(y_j, C_j(\overline{y})$, containing the service parameters for different storage technologies. The idea behind the DP approach is that, the optimization problem defined in Eqn. (3.7a) is divided into independent sub-problems, where for each IoT application, the storage

technology is chosen separately and the filled 2-D table is used to solve the future sub-problems (HC & et al, 2009). With the increase in the table entries, algorithm leads to efficient decisions.

Algorithm 16 summarizes the heuristic design to approximate a solution to the optimization problem formulated in Equation. (3.7a). The algorithm starts with $\overline{y} = 0$ and $T(y_j, C_j(\overline{y})) = 0$, and iteratively chooses the storage technology for $m$ IoT applications data types. The algorithm calculates the dynamic parameters cost for $j^{th}$ storage technology at line 5. Given the IoT application service requirements, algorithm checks on line 7 if it matches with the calculated cost parameters. In case of match, at line 8, the same storage solution is assigned to the current IoT application data type. Once the storage is selected, the cost parameters are stored in the table for future use. Then, for the next IoT application, the algorithm checks the table at line 11, if any storage technology fulfills the current service requirements.

**Time Complexity:** For the time complexity of Algorithm. 16, the loop in line 4 runs at most $i$ times for all the IoT applications. Similarly, the loop in line 5 runs $j$ times in a worst case scenario for all the storage technologies. In addition, the line 6 runs for the $k$ number of service requirements. Thus, the time complexity of Algorithm. 16 is $O(ijk)$, which is polynomial-time.

**Greedy-style (GS) heuristic:** In some IoT applications scenarios, the owner may only be interested in few important service requirements. For instance, privacy may be the only service requirement in IoT health application. Therefore, in this case, calculating all the cost parameters of storage technologies is not feasible. Therefore, the proposed algorithm only calculates the service parameters of the storage technology associated with the IoT application service requirement. The main rationale behind the GS heuristic is that the algorithm selects the storage technology using first fit strategy, as soon as certain number of defined optimization conditions (Equation $(3.7c) - (3.7g)$) are fulfilled. Algorithm 3.2 summarizes the GS heuristic algorithm.

In this algorithm, we define $\gamma$, which corresponds to the service requirements of the IoT applications, as the conditions (Equation $(3.7c) - (3.7g)$), that needs to be fulfilled before selecting a storage technology. For instance, if the owner has only defined availability and price requirements, the $\gamma$ would be defined as a 2 parameters tuple, i.e., $<availability, price>$.

At line 6, the associated cost parameters of storage technologies are calculated for each IoT application data type $m$. Line 7 matches these cost parameters with the $\gamma$ service requirements of IoT application data type $m$. At line 8, the storage solution is selected for IoT application $m$ in line 9, if the $\gamma$ conditions meet.

Algorithm 3.2 Greedy-style heuristic

```
1  Initialize M
2  Initialize m ← 1
3  Initialize γ ← γ1, γ2, ..., γm
4  while m ≠ M do
5      for y ← 1 to n do
6          Find γm parameters costs associated with yn Match parameters costs with γm
             Equation (3.7c) − (3.7g)
7          if γm conditions are fulfilled then
8              Assign yn storage technology to mth IoT application
9              break
10         end if
11     end for
12     m = m + 1
13 end while
```

**Time Complexity:** For the time complexity of Algorithm. 3.2, while loop runs for $m$ times, while the for loop runs for n times in the worst-case scenario. In steps 7, 8, and 9, the calculation takes O(1) time. Thus, the worst-case overall computational complexity of Algorithm 3.2 is $O(mn)$, which is polynomial-time.

**Temporal dependencies:** The parameters that are fed in the middleware, e.g., price, performance, etc., are calculated based on the information provided by the storage technologies. Any uncertainty in these parameters can lead to SLA violations. The problem of SLA violation identification and compensation can be solved in future. Moreover, the application and storage parameters can be changed with time, and our proposed algorithms do not consider these temporal dependencies. It might be possible that a storage technology is selected for an IoT application but after a certain time, it violates the QoS parameters. This problem can be considered in future to accommodate the time dependency factors in the proposed algorithms.

**Discussion:** To decide the storage solution, it is not mandatory to consider all the service requirement for a particular IoT application and the storage selection depends on the service requirements associated with the IoT application. For instance, if the minimum budget requirement is specified by the owner, only constraints (3.7c) and (3.7d) will be considered in the optimization problem (3.7a). We argue in the favor of GS heuristic in this case because the algorithm allows the individual IoT applications to specify their own service requirements. However, in case of all service requirements, it is recommended to use the DP-based heuristic to find a storage technology for IoT applications because of its accuracy and performance as described in Section 3.6. It should be noted that this work does not improve the performance of the proposed design presented in (Danish & et al., 2020). This work extends our previous studies with the technical details of neural network based intelligence and adaptive aggregation rate.

In this work, we propose the use of blockchain-based middleware, however the economic model of the middleware design has not been explored, i.e., who is going to pay for the blockchain transaction cost. We consider the middleware design as a third-party solution running on fog node. The blockchain transaction cost can be incorporated in the subscription fee however, we consider the economic model of middleware design as an open problem which we will address in future work.

## 3.4 Neural network-based maintenance strategy

Our proposed middleware design is shown in Figure 3.2, which consists of a core middleware module with three sub-modules, i.e., decision module, neural network module and feedback module. In the decision module, we are making the storage selection decisions with the help of blockchain-based data placement decision protocol, by solving the decision optimization problem formulated in Equation (3.7a), subject to IoT application's service requirements. Once the decision has been taken by decision module and a storage technology has been selected for IoT applications at time $t$, the middleware has now entered in the dynamic phase, i.e., the service requirements of IoT applications and service parameters of storage technologies are changing dynamically according to a certain pattern. Selecting the storage technology again

for the next time continuously would incur significantly high computation and communication overhead. Moreover, as the parameters are published on the blockchain network before making a data placement decision, therefore, continuously publishing the storage and IoT parameters would result in high blockchain transaction and storage overhead. Thus, there is a need to intelligently select the time $t + 1$, i.e., when to re-make the data placement decision to select the storage technology, by taking into consideration the dynamic service requirements and service parameters of the storage technologies. Therefore, the main question is, what should be the maintenance strategy for the specific IoT application, i.e., when to re-run the blockchain-based data placement protocol again to decide the new storage solution for the IoT application?

To answer this question, we introduce a neural network module in our architecture, which learns the dynamically evolving characteristics of the IoT applications and storage technologies to intelligently define the maintenance strategy. The module considers the conclusions drawn from the set of available features and enables automatic maintenance configuration for the large-scale IoT data through neural network algorithms. Our neural network approach accomplishes the following task:

- The module employs long short-term memory (LSTM) to predict future IoT data patterns and schedule at what the time the middleware should re-evaluate the data placement decision.

The neural network module requires a set of features in order to train and predict a target, which is the maintenance configuration in our case. To decide when to re-evaluate the data placement decision depends on multiple factors, including the IoT applications service requirements, i.e., amount of data generated, number of sensors, data retrieval pattern etc. and service parameters of storage technologies. Based on these samples, a neural network model is built to classify the maintenance configuration of the data placement decision.

### 3.4.1 Features extraction

We now explain the features considered in our proposed model.

- *Performance:* We construct a feature vector $f_{per}$ = ($lat$, $ava$) from the storage technology parameters, where $lat$ and $ava$ are the latency and availability of the storage technologies. These set of features dynamically change for a given storage technology.
- *Pricing:* We construct a feature vector $f_{pri}$ = ($storet$, $bwpri$) from the storage technology parameters, where $storet$ and $bwpri$ are the storage/retrieval price and bandwidth price requirement for given IoT application's data sample respectively.
- *Data generation:* We consider a feature vector $f_{sdata}$ = ($size$, $devices$, $dur$), which corresponds to the data size, number of devices and duration respectively. These set of features are changed dynamically for different IoT applications with a given data sample. As these features are highly correlated with each other, we consider data size as a feature in our proposed model.
- *Data retrieval:* We consider a feature vector $f_{rdata}$ = ($size$, $freq$), which corresponds to the retrieval data size and frequency respectively. These set of features are different for different IoT applications with a given data sample. We consider this feature for our model because few storage technologies also incur data retrieval cost.

We now explain the proposed neural network-based maintenance strategy for our proposed middleware design.

### 3.4.2 Pred-ms: lstm-based maintenance strategy

The maintenance strategy decisions can be made on-the-fly after a certain time period however, storing the IoT applications service requirements and storage technologies parameters after every time sample $t$ can result in high transaction and storage overhead on the blockchain network. Therefore, instead of deciding the maintenance strategy on-the-fly, the IoT data sample and service parameters can be predicted ahead of time to schedule the maintenance strategy decisions intelligently. Thus, we propose Pred-MS, which schedules the maintenance strategy ahead of time by predicting the future IoT data sample and service parameters.

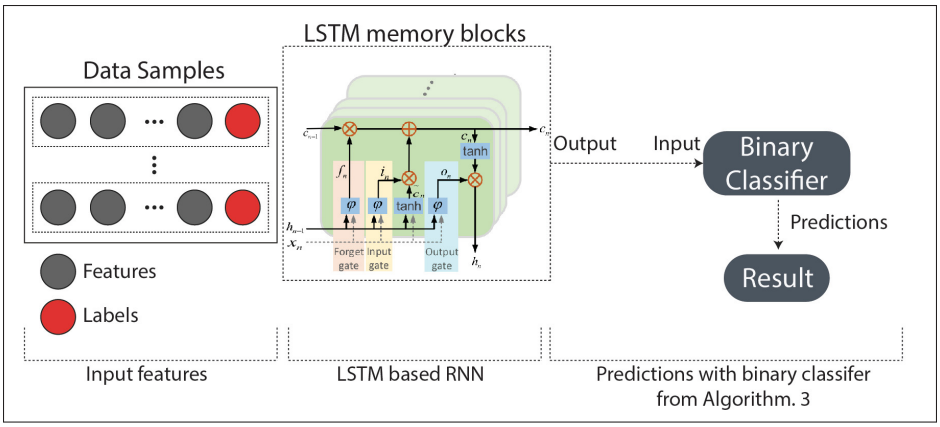



Figure 3.3 Lstm-based binary classification (pred-ms)

We consider the IoT data sample and associated service parameters as a time series data and propose a Long Short Term Memory (LSTM)-based Recurrent Neural Network (RNN) to predict the IoT data sample and service requirements. The results are fed as an input to the binary classification model to schedule the maintenance strategy, i.e., to schedule the re-evaluation of the data placement decision for the predicted IoT data samples. This model is different from the previous model in the sense that model decides the maintenance strategy on the fly based on the input parameters however, Pred-MS first predicts the IoT data samples with associated parameters and schedule the maintenance strategy decision in advance. To sum up, the problem can be formally defined as follows.

**Given:**

- A collection of different IoT applications service requirements over a period of time.
- A collection of storage technologies service parameters.

**Goals:**

- We first aim to predict the IoT data samples and service requirements in the future based on the current information. The binary classifier then predicts the probability of the selection of a better storage technology for each predicted IoT data samples and their service requirements.

The selection of LSTM based-RNN network for the considered problem is because of its ability to hold long-term memory. As we considered IoT data sample and associated service parameters as a time series data therefore, it is possible that $n^{th}$ sample in the sequence can be influenced by an input that was given many time steps before. Thus, we argue in the favour of LSTM based RNN as they are good for holding long-term memory. Figure 3.3 shows the overview of the proposed model with the LSTM-based RNN and binary classifier. First, the already available IoT data samples and service requirements are fed into the LSTM to predict future data samples and service requirements. Let the time series data is represented by $\mathbf{x} = \{x_1, x_2, ..., x_n\}$. The LSTM has an input gate $i_n$, which decides what new information needs to be stored by the LSTM in the new cell state. Input gate can be represented as:

$$i_n = \varphi(b_i + u_i^T x_n + w_i^T h_{n-1}) \tag{3.10}$$

where $x_n$ is the input sequence of IoT data samples and service requirements, where $h_{n-1}$ is the output vector at previous time step. Also, $w_i$, $u_i$, $w_f$ and $u_f$ are the recurrent and input weights, respectively. Moreover, forget gate $f_n$ decides what information needs to be kept or thrown away. Forget gate can be represented as:

$$f_n = \varphi(b_f + u_f^T x_n + w_f^T h_{n-1}) \tag{3.11}$$

where $f_n \in [0, 1]$, i.e., the value of 1 allows the LSTM to keep the last state, while the LSTM forgets the previous state for $f_n = 0$. The memory of the LSTM $C_n$ is updated by adding new memory content $\hat{C}_n$ and by forgetting current memory by the factor $f_n$ as shown in Equation (3.12):

$$C_n = f_n C_{n-1} + i_n \hat{C}_n \tag{3.12}$$

$$\hat{C}_n = \tanh(u_v^T x_n + w_c^T h_{n-1} + b_c) \tag{3.13}$$

where tanh is the activation function. Finally the output of LSTM $h_n$ is given by:

$$h_n = o_n \tanh(C_n) \tag{3.14}$$

$$o_n = \varphi(b_o + w_o^T x_n + u_o^T h_{n-1}) \tag{3.15}$$

where $o_n$ is the output gate, while $h_n$ signifies the predicted IoT data samples and the associated parameters. Once the IoT data samples and service requirements are predicted, they need to be fed as an input to the binary classifier to predict when is the next time, there is a need to re-evaluate the data placement decision.

We also consider a neural network based binary classification model, which decides whether to re-evaluate the data placement decision for current IoT data sample.

The problem is defined as follows:

**Given:** A collection of different IoT applications service requirements and storage technologies service parameters predicted by LSTM-based RNN network over a period of time.

**Goal:** We aim to learn a classifier to determine whether to re-evaluate the data placement decision for the given IoT applications service requirements. Thus, based on the new IoT data sample and the previously used storage technology, the classifier predicts the probability of selection of a better storage technology for the given IoT data sample and its service requirements.

In our work, we employ supervised learning technique by manually labeling the data offline, where $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{V} = \{0, 1\}$ are the input and output spaces, respectively. $\mathcal{X}$ is a feature representation for the $i^{th}$ current IoT data sample, while $v_i \in \mathcal{V}$ is the class label. $v_i = 1$ if there is a need to re-evaluate a data placement decision and $v_i = 0$ otherwise. We provide the detailed description of the dataset in the next subsection. Mathematically, the binary classification problem can be formulated as Equation (3.16):

Algorithm 3.3 Pred-ms lstm-Based binary classification model

```
1  Initialize Model
2  Initialize the time series IoT dataset
3  Train the binary classification model
4  while true do
5      Predict h_n for k samples in the future for each k do
6          Input h_n in binary classification model trained at line 3
7          if v_i == 1 then
8              Re-make the data placement decision for k^th sample
9          end if
10         if v_i == 0 then
11             Pass
12         end if
13     end for
14     Go back to step 5
15 end while
```

$$v_i = \begin{cases} 1 \;\; if \;\; p(v_i|\mathcal{X}, Model) \geq T \\ 0 \;\; otherwise \end{cases} \tag{3.16}$$

where $T$ is the probability threshold and represents the trade-off between the number of false positives and false negatives.

Algorithm 3.3 summarizes the LSTM-based maintenance strategy model. In this model, a binary classification model is first trained with the data samples obtained by solving the decision optimization problem, formulated in Eqn. 3.7a. At line 5, the LSTM-based RNN is used to predict $k$ IoT data samples along with the service requirements in the future. Once, the $k$ data samples are predicted, the output of the LSTM model at line 5 is fed to the binary classification model one by one at line 7. The binary classification model makes the prediction about these input data sample $h_n$ at line 7.

### 3.4.3 Dataset description

In this work, we have used the TimeScaleDB bench-marking tool[1] to generate the large-scale IoT data. This tool is meant to simulate the data load in an IoT environment. It considers the supply chain IoT applications and simulates the data streaming from a set of trucks belonging to a trucking company. The tool allows to generate the IoT data with different number of trucks, sensors, generation time interval etc. By utilizing these parameters, we generated a time series data with a time sample $t$ of 1 hour. These generated data parameters include the number of trucks, the data generation time and the total amount of data generated for the period of 1 hour. Moreover, the IoT applications service requirements are generated randomly for each instance $t$ (Danish & et al., 2020). Our compiled dataset consists of 16000 $t$ instances without duplicates. Each row indicates the number of trucks, the generation time interval, total amount of data generated along with the IoT applications service requirements, storage parameters and the previously selected storage at instance $t-1$.

**Data Labels:** In this work, we employ supervised learning technique to manually label each data sample offline. Based on these IoT application service requirements and the data sample information at time $t$, we first approximate a solution to the decision optimization problem formulated in Section 3.3 to find the storage technology and label the first row as 1. Now, for the next data sample, we use new data sample to find the storage technology and compare the result with previous one. If the selected storage technology at $t$ is different from the one at $t-1$, we label the instance as 1 and 0 otherwise. This labelled data is then used to train the neural network model explained in this section before.

## 3.5 Adaptive aggregation rate

The IoT sensor nodes generate a large amount of raw data which is pre-processed and aggregated by the fog nodes before storing it in the selected storage technology. Fog nodes extract the time-critical IoT data and forward the archived data to cloud or multi-cloud premises for future

[1] https://github.com/timescale/tsbs

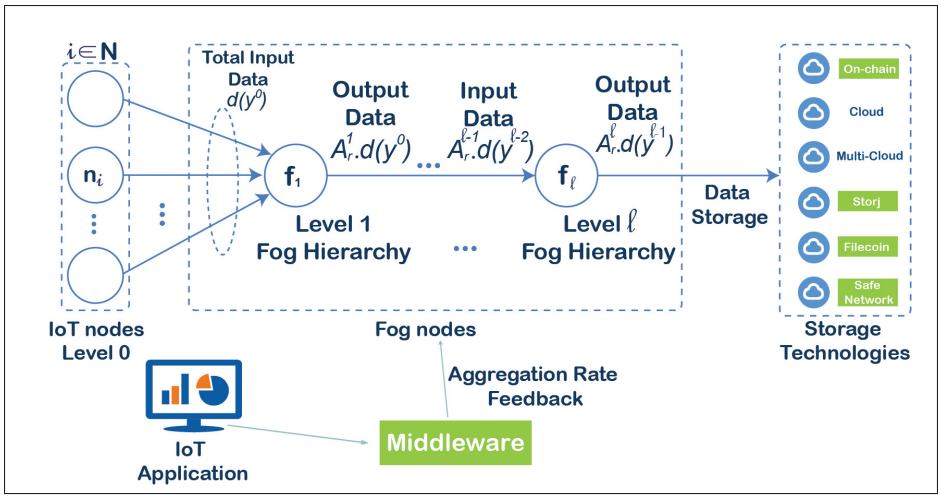



Figure 3.4 Hierarchical fog architecture

storage and retrieval. A fog node is a smart edge computing device, which is context-aware and can smartly tune itself based on application feedback (Aazam & Huh, 2016).

BlockAIM exploits this feature by providing aggregation rate feedback to the fog node to optimize the overall price of the data storage while maintaining sufficient data quality for a given IoT application. Once BlockAIM has performed its selection, the feedback loop adapts the aggregation rate of the IoT application to minimize the overall price of IoT data storage.

Our proposed solution operates in a hierarchical fog architecture with multiple levels as shown in Figure 3.4, where level 0 (L0) is comprised of the IoT nodes (Imai & et al., 2018). We denote the set of IoT nodes $N = \{n_1, n_2, n_3, ..., n_M\}$, where M is the total number of IoT devices, which send sensed data to the level 1 (L1) gateway fog node for aggregation and pre-processing. Let $Y^j$ denotes the set of hierarchical logical fog nodes at level $l$ and $Y^j = \{y^j \mid j = l\}$ for $l$ = 1,2,...,$L$, where $y^j$ denotes the $j^{th}$ logical node at level $l$. We denote the aggregation and compression capability of hierarchical fog nodes by $A_r = \{A_r^1, A_r^2, A_r^3, \ldots, A_r^l\}$, where $A_r^l$ denotes the aggregation rate of $l^{th}$ level hierarchical fog node. It should be noted that one level of hierarchical fog node processes the IoT application data at a time, and thus the next hierarchical fog node level will start aggregating the same data after it has been finished by the previous level of hierarchical fog node.

Let $d(y^0)$ denotes the raw data generated by the set of $N$ level 0 IoT nodes and is sent to level 1 gateway fog node for aggregation. Level 1 gateway fog node aggregates the data with aggregation rate $A_r^1$ and forwards the data $d(y^0) = (1 - A_r^1)\ d(y^0)$ to level 2 fog nodes, where $A_r^1 \in \{0, 1\}$. A hierarchical fog node $y^j$ at some level $l$ receives the data $d(y^{l-1})$ from lower level hierarchical fog nodes. When the hierarchical level is 1, i.e., $l = 1$, the gateway fog node receives raw IoT data from IoT devices at level 0. When $l$ >1, it receives the aggregated data from its precedent fog node $y^{l-1}$. When the hierarchical fog node at level $l$ receives the data from its precedent fog node $l - 1$, it aggregates the data with the rate $A_r^l$ and forwards it to the hierarchical fog node at level $l + 1$. The total amount of data received by hierarchical fog node at level $l$ can be represented as Equation 3.17:

$$d(y^l) = \begin{cases} d(y^0), & \text{for } l = 1 \\ \prod_{i=1}^{l} (1 - A_r^i) d(y^{l-1}), & \text{otherwise,} \end{cases} \tag{3.17}$$

where $d(y^l)$ is the total IoT data received from level $l - 1$ fog node. Moreover, the total aggregation rate can be derived from the individual aggregation rates of all the hierarchical fog nodes and can be represented as Equation 3.18:

$$A_r^{total} = \prod_{i=1}^{l} A_r^i, \tag{3.18}$$

As the data will be aggregated on multiple hierarchical fog nodes before reaching the final level $l$ fog node, the total amount of data generated by the level $l$ fog node can be represented as Equation 3.19:

$$d^{total} = d(y^0) \prod_{i=1}^{l} (1 - A_r^i), \tag{3.19}$$

If $P$ is the price to store the aggregated data in the current storage technology, the overall cost $C$ can be represented as:

$$C = P d(y^{total}), \tag{3.20}$$

It can be seen from Equation (3.17) and (3.20) that the total aggregated data size $d(y^{total})$ decreases by increasing the value of $A_r^{total}$, results in small data size and low overall cost. However, the increased aggregation rate results in low data precision and quality. In contrast, the data quality increases by decreasing the value of $A_r^{total}$, which results in increased data size and consequently high storage cost. Therefore, the overall storage cost and the data quality of the aggregated data depends on the aggregation rate. If $P$ is given, there is a need to find an optimal value of aggregation rate $A_r^{total}$ such that the overall cost is minimized. More formally,

$$\min_{A_r \in [0,1]} \quad C \tag{3.21a}$$

$$\text{subject to} \quad A_r^{total} \leq T_q,\ 0 \leq A_r \leq 1. \tag{3.21b}$$

$$C \leq C_{max}. \tag{3.21c}$$

$$d(y^{total}), P, l > 0. \tag{3.21d}$$

The constraint (3.21b) states that the aggregation rate should not exceed the predefined data quality threshold $T_q$, while the constraint (3.21c) states that the overall cost should not exceed the current cost $C_{max}$ of the storage technology. The constraint (3.21b) is set by the application owner for a specific IoT application and fog nodes keep track of this information in order to aggregate the IoT data. Fog node sends the updated data quality threshold information to the middleware's feedback module. Moreover, the information for the constraint (3.21c) is retrieved from the data placement module of the middleware design.

This work considers a hierarchical fog architecture with multiple virtual fog nodes in order to generalize the formulated optimization problem. However, if we already know the data quality threshold and the cost, it is possible to solve this optimization problem with a single fog node, instead of considering several stages.

## 3.6 Evaluation and results

In this section, we evaluate and analyze the efficiency, accuracy and viability of the proposed middleware design. In addition, we also analyze the blockchain overhead aspect of the proposed protocol.

### 3.6.1 System setup

We run our experiments on UBUNTU 16.04 desktop with Intel $Core^{TM}$ i7-8650U processor ($8^{th}$ generation) and 16 GB RAM. We use the same machine to label our data offline and Google Colab is used to train the model in our work.

**Simulation Parameters:** Currently, the decentralized storage technologies are in their infancy and the associated service parameters of these storage technologies are not available publicly. Therefore, because of the public availability, in our experiments, we only employ the real service parameters of storj and cloud. In our experiments, we have considered Google cloud with 0.026 $/GB data storage cost and 0.085 $/GB bandwidth cost. The availability cost of Google cloud is set to 99.95%, while the privacy cost is set to 0, i.e., Google cloud offers no privacy. Similarly, per GB data storage cost for Storj is set to 0.015 $/GB[2], while the bandwidth cost is set to 0.03 $/GB[3]. The data retrieval cost for Storj is set to 0.05$/GB. Similar to Google cloud, the availability cost is set to 99.95%[4], while the privacy cost is set to 0. The block time is set to 0 and the link bandwidth is set to 100 Mbps for both storage technologies. Moreover, the IoT applications service requirements are generated randomly in these experiments.

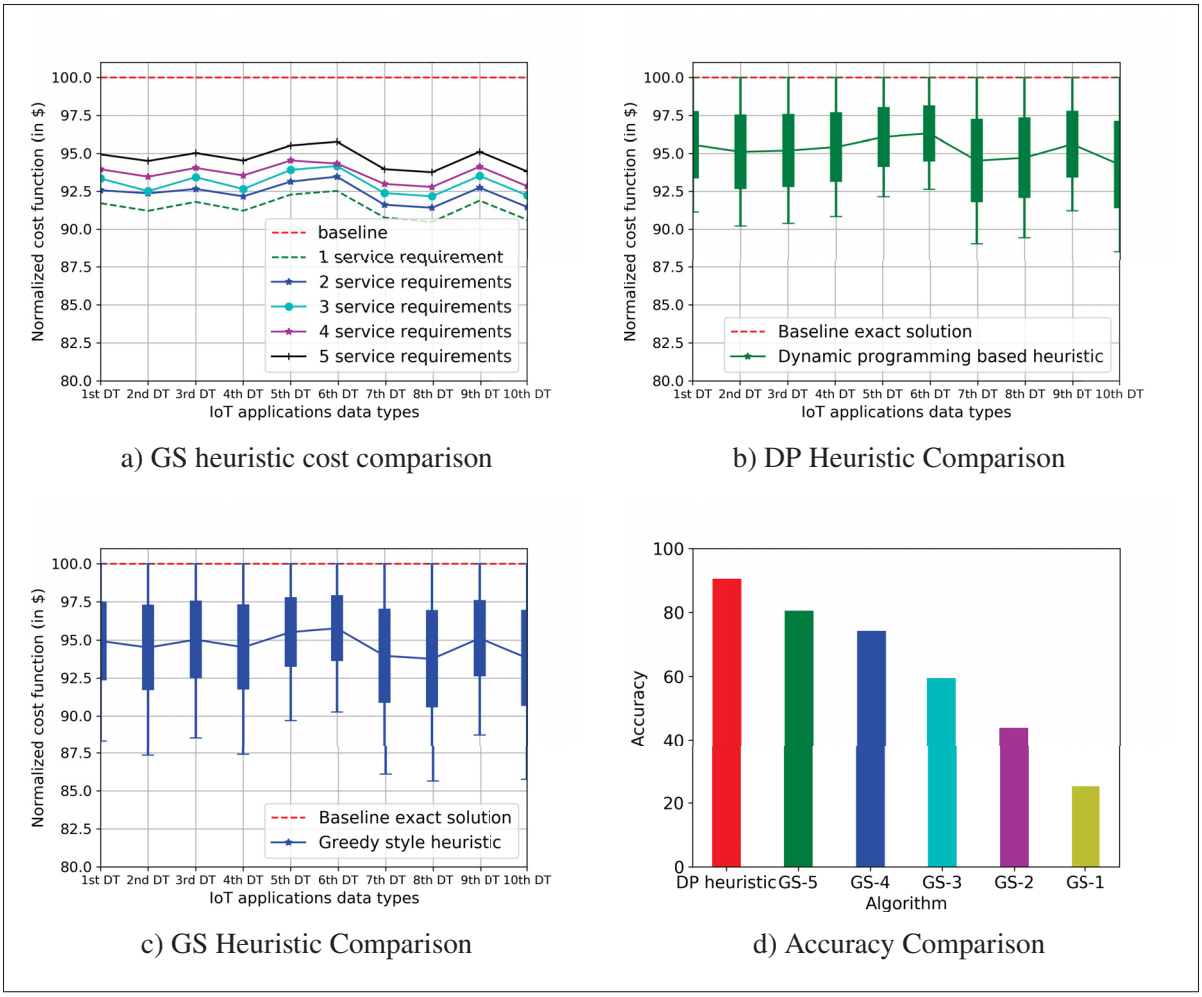
### 3.6.2 Results for data placement

**Cost Comparison:** Firstly, we compare the middleware data placement decision for two proposed heuristic algorithms with baseline. The IoT applications service requirements are

[2] https://storj.io/

[3] https://storj.io/storage-node-estimator/

[4] https://storj.io/blog/2019/08/the-role-of-qualification-gates-in-getting-to-beta-and-beyond/




a) GS heuristic cost comparison

b) DP Heuristic Comparison

c) GS Heuristic Comparison

d) Accuracy Comparison

Figure 3.5 Heuristics comparison

generated randomly and the solution to the decision optimization problem is approximated through the real-world service parameters of multi-cloud, cloud and storj. To generate acceptable average comparison results, each simulation is repeated for at least 1000 times. A normalized cost function ($\frac{approximatedsolution}{optimalsolution} * 100$) is used to compare the performance of heuristics.

Figure 3.5a compares the GS heuristic cost with the baseline for ten IoT applications. It can be seen that, considering all the service requirements, the overall normalized cost is close to optimum. However, decreasing the considered service requirements increases the GS heuristic algorithm cost. This is because, for few service requirements, the decision will be taken by the algorithm once the required IoT application's service requirements are fulfilled thus, resulting

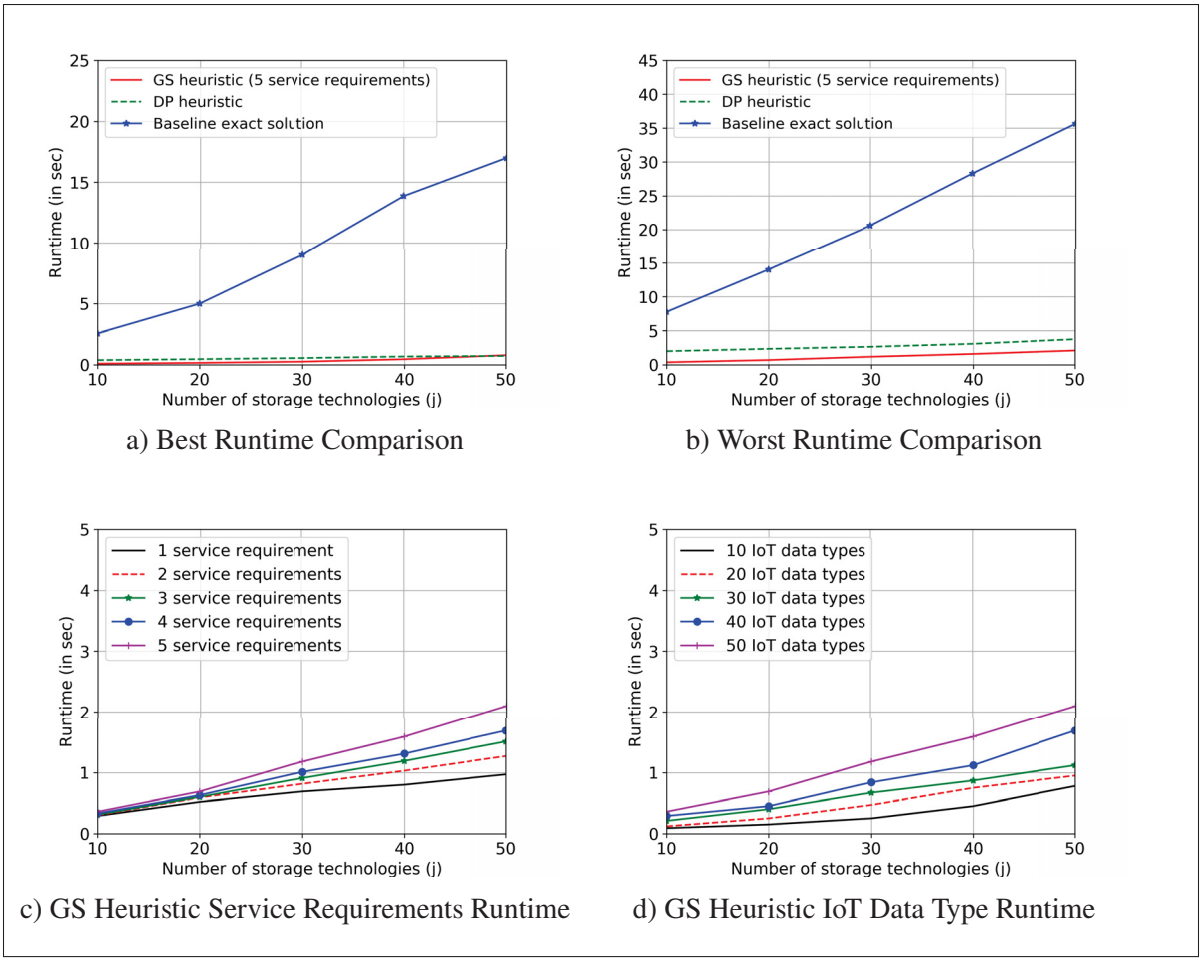



Figure 3.6 Heuristics comparison

in more cost. It can be observed in Figure 3.5b and Figure 3.5c that the cost incurred by DP heuristic outperforms the GS heuristic and is close-to-optimum exact solution. This is due to the fact that, after the completion of the cost table, the storage technology with minimum associated cost will be selected by the DP heuristic, compared to random selection of storage technology in GS heuristic.

**Accuracy Comparison:** We now evaluate the accuracy, i.e., the exact decisions made by the heuristics, for the two proposed heuristic algorithms in Figure 3.5d. We calculate the algorithms' accuracy by comparing the data placement decisions for the given service requirements with the optimal placement decision. Experimental evaluation demonstrates that the DP heuristic

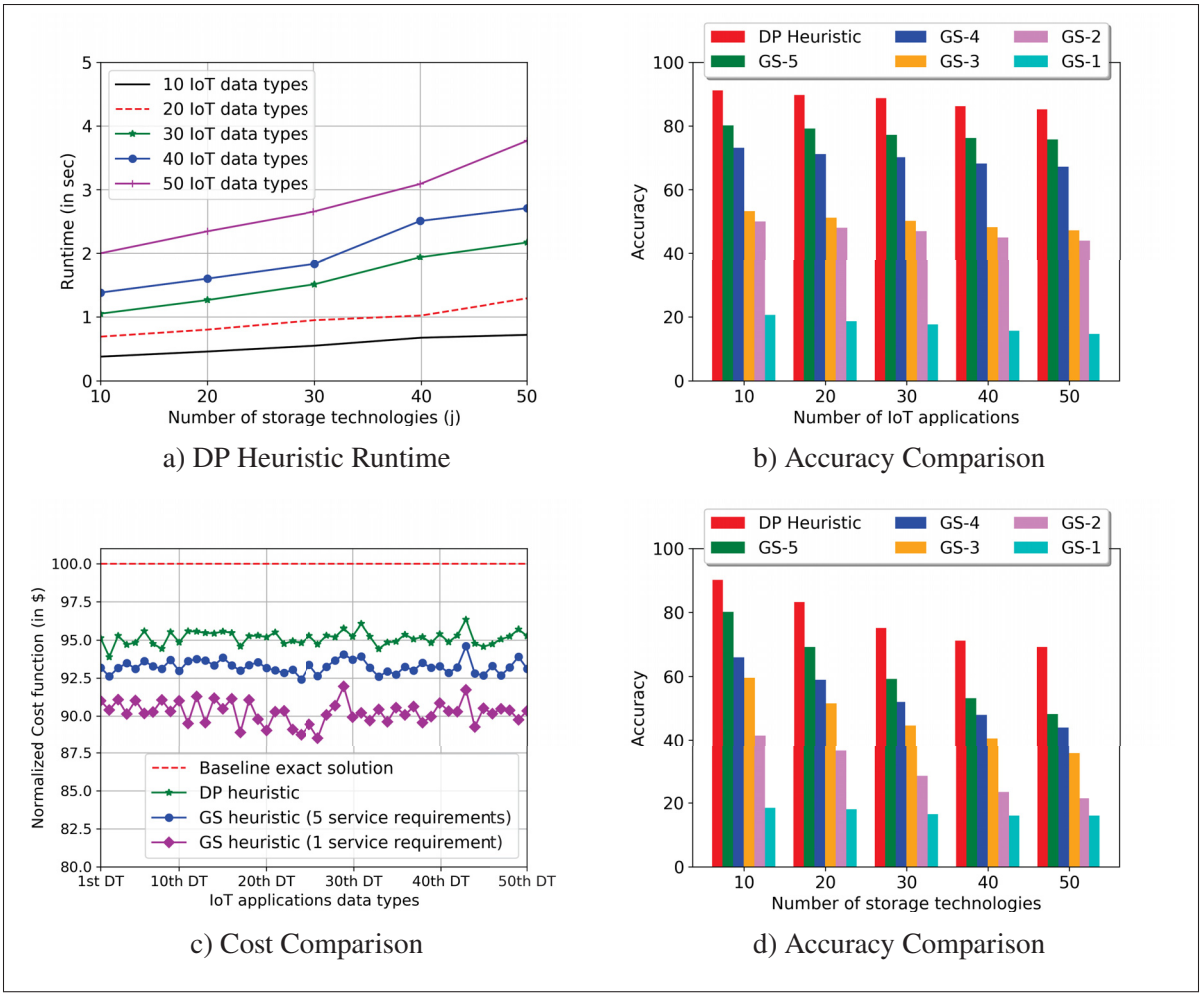



Figure 3.7 Heuristics comparison

provides almost 92% accuracy compared to the GS heuristic, which achieves almost 80% accuracy. For fewer service requirements, the accuracy of greedy-style heuristic is low. However, the IoT applications' service requirements are always fulfilled by the algorithm, even with the low accuracy.

**Algorithms Runtime:** We further compare the algorithms runtime in terms of best and worst case performance with the baseline exact solution in Figure 3.6a and 3.6b respectively. The baseline solution is an exhaustive search solution, which chooses the optimal storage technology by going through all the possible solutions to the optimization problem defined in Eqn. 3.7a. For the best case scenario, we consider 10 IoT applications in the proposed algorithms, while we

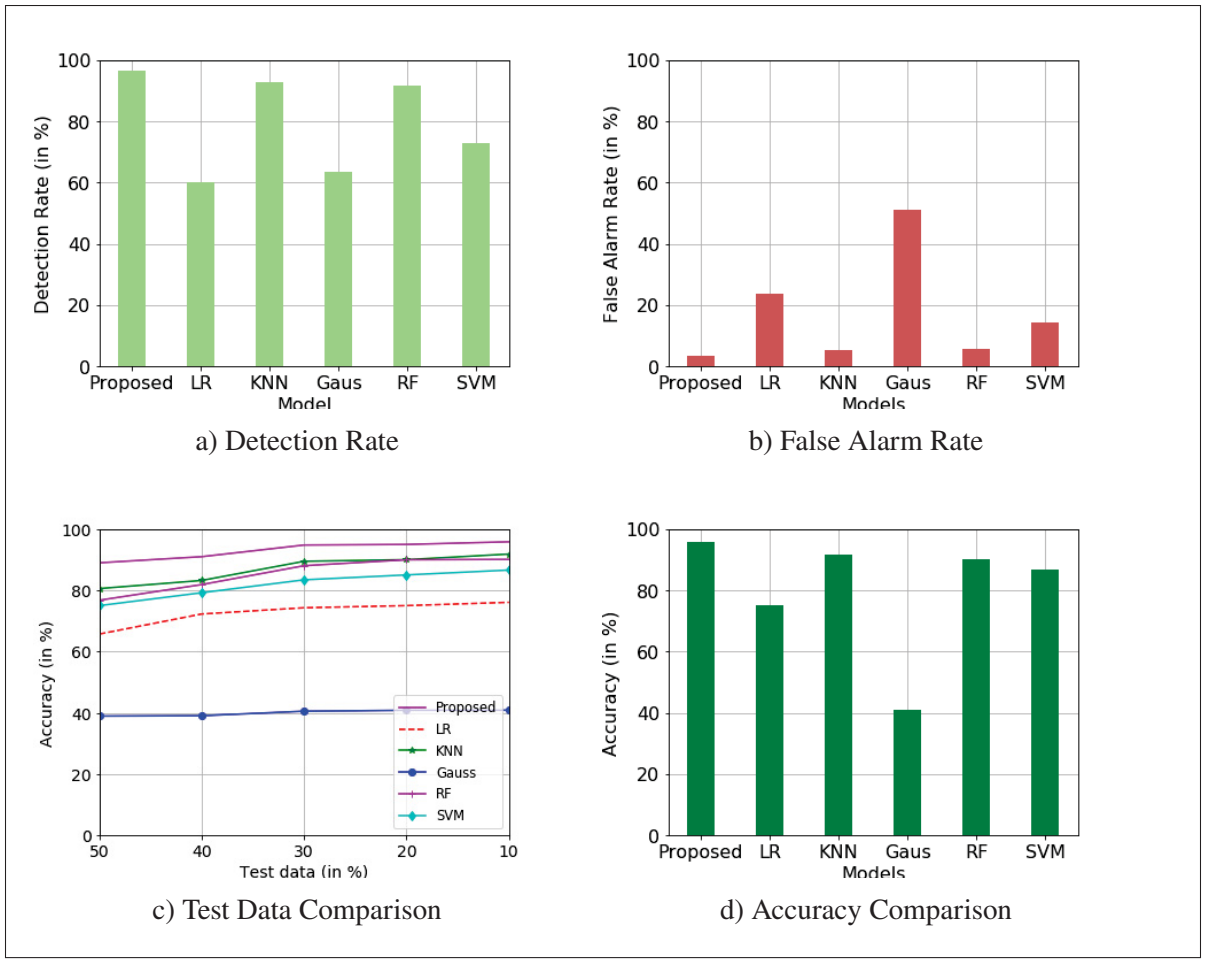



Figure 3.8 Maintenance strategy analysis

consider 50 IoT applications to perform the algorithm comparisons for the worst case scenario. In both best and worst case scenario, the GS heuristic algorithm gives better performance and outperforms the DP heuristic algorithm. However, it achieves poor accuracy compared to DP heuristic as shown in Figure 3.5d. Furthermore, compared to the exact solution to the optimization problem, the runtime is significantly reduced for the proposed algorithms.

**Cost Comparison:** In order to investigate the worst case performance of the proposed algorithms, we perform the sensitivity analysis for high workload. To perform these experiments, we generate the storage technologies parameters randomly between the maximum and minimum values of the real world parameters of multi-cloud, storj and cloud. We perform these experiments many

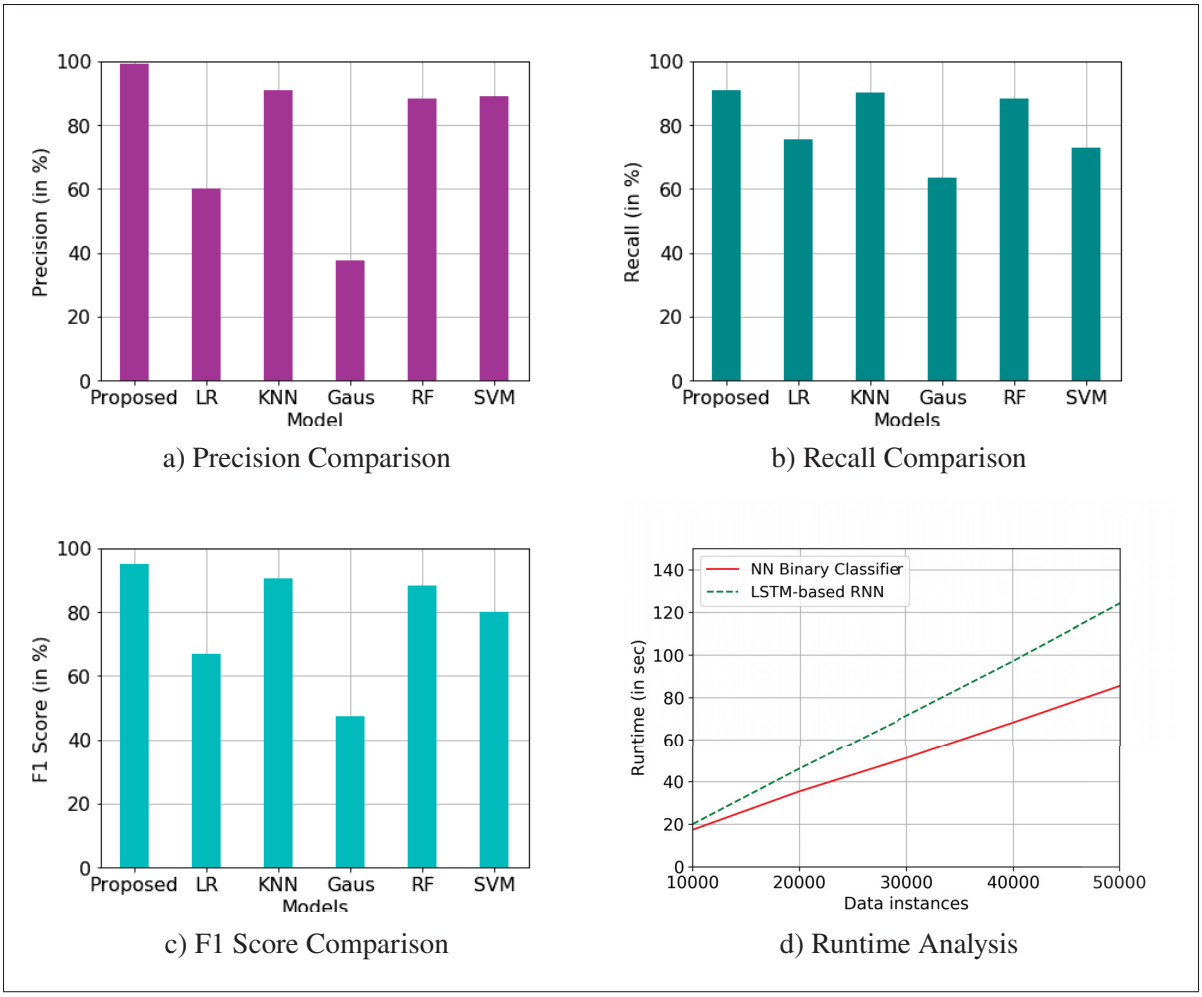



Figure 3.9 Maintenance strategy analysis

times to average out the results. Figure 3.7c compares the cost of optimization for the proposed algorithms against the baseline exact solution. This is because, the probability of the selection of a sub-optimal storage increases with the increase in storage technologies, while fulfilling the IoT application requirements.

**Accuracy Comparison:** Figure 3.7b compares the two proposed heuristic algorithm in terms of accuracy for different IoT application data types. we can see that the accuracy is not affected with increase in the number of IoT applications. This is because, for each IoT applications data type, the storage decision is taken individually. However, the runtime of the proposed algorithms increases with the increase in the number of IoT applications. Figure 3.7d illustrates

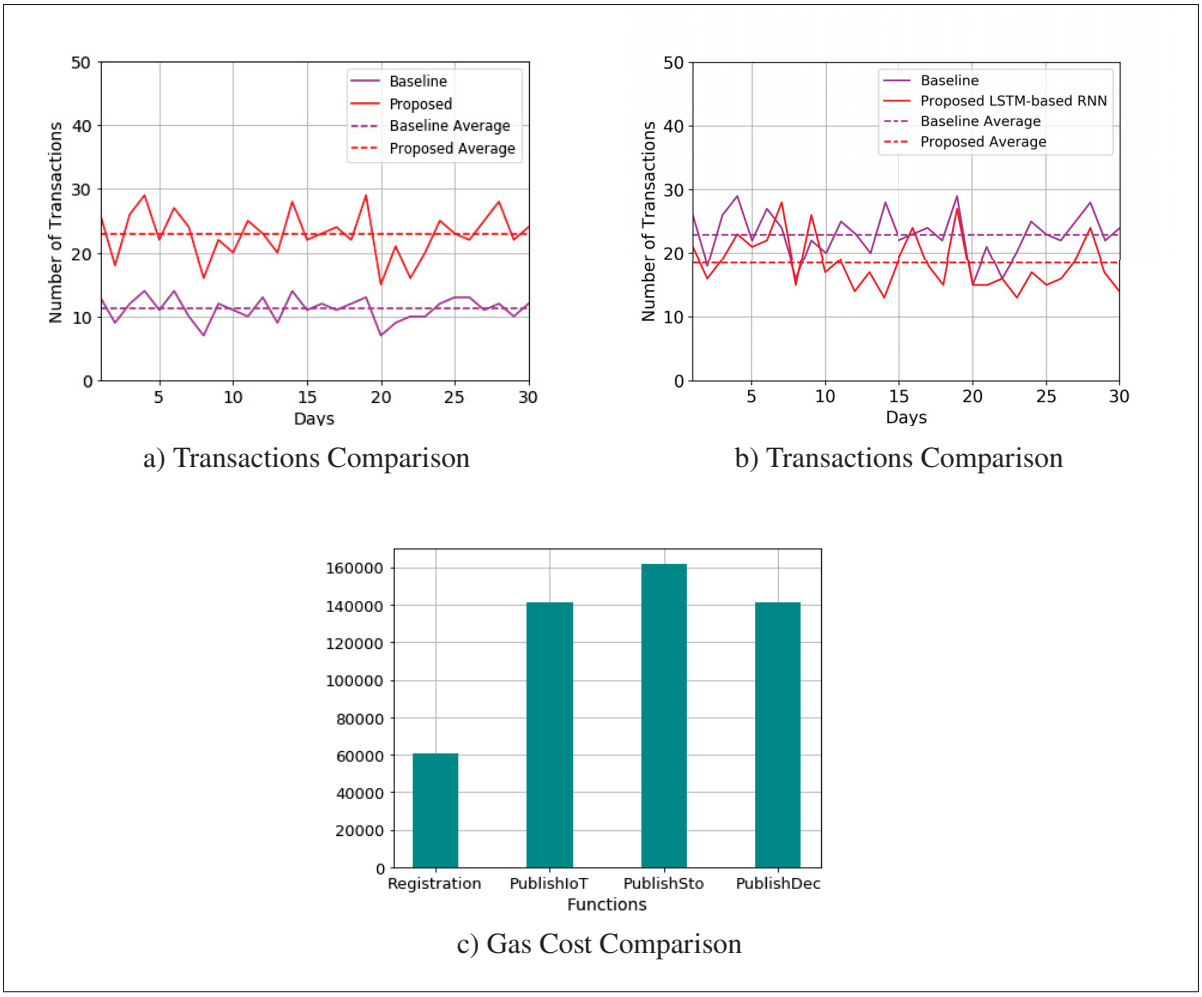



Figure 3.10 Blockchain analysis

the algorithms accuracy for large number of storage technologies with randomly generated service parameters. It can be seen that the accuracy of proposed heuristics decreases for large number of storage technologies. This is because, the probability of selection of a sub-optimal storage with associated high cost increases with the increasing number of storage technologies, while fulfilling the IoT application requirements.

**Computation Time:** We investigate the sensitivity analysis for the average runtime, i.e., the algorithm runtime for large input instances. Figure 3.6c and 3.6d illustrate the time consumed by CPU for GS heuristic for deciding the storage technologies for different IoT application data types. We can see that increasing the number of IoT application data types and associated service

requirements, the runtime of GS heuristic increases. This is due to the fact that for large input, the algorithm is required to search through a large solution space, thus leading to high runtime. Similarly, we can see large runtime for DP heuristic algorithm in Figure 3.7a for multiple IoT data types.

### 3.6.3 Results for the maintenance strategy

In this section, we first analyze the binary classification of the IoT data sample along with its service requirements. Our module uses a neural network-based binary classification model to accomplish this task. We use $keras$ library with $tensorflow$ as a backend to run our experiments. We use the following metrices in order to evaluate the performance of the proposed model: Accuracy, F-Score, Recall and Precision. Moreover, we also compare the performance of our proposed binary classification model with other traditional machine learning classification models: Logistic Regression (LR), Support Vector Machines (SVM), Random Forest (RF), Gaussian Naive Bayes (Gaus), K-Nearest-Neighbours (KNN).

**Parameters Settings:** We split our dataset randomly with a ratio 75:25, resulting in 75% training data and 25% testing data. We consider a two layer neural network model with 10 neurons each. Moreover, we run the model several time independently to achieve the average performance metrics. The objective function is minimized by using the Adam optimizer and the batch size is set to 32. Finally, we use k-fold cross validation scheme in our experiments to avoid over-fitting of the neural network model with k=5.

Figure 3.8a and 3.8b show the comparison of the proposed algorithm with traditional classifiers in terms of detection rate and false alarm rate, respectively. It can be seen in the Figure 3.8a that the proposed neural network based binary classification algorithm outperforms other classification algorithms and achieves almost 96% detection rate. Furthermore, we can see that the proposed algorithm achieves minimal false alarm rate among all the classifiers. Figure 3.8c shows the accuracy comparison of the proposed algorithm with the traditional classifiers with varying test data set. It can be seen that for the 50% test data and 50% training data, the accuracy of the

proposed algorithm is low. However, decreasing the test data to 10% yield better accuracy. This is because the model learn better with more training data to perform well on the test dataset.

Figure 3.8d, 3.9a, 3.9b and 3.9c show the comparison of the proposed algorithm with traditional classifiers in terms of accuracy, precision, recall and f1 score, respectively. It can be seen that the proposed classification algorithm outperforms all the traditional classifier. This is because the structure of the neural network captures the nonlinear relations among the features in the dataset, thus leading to better performance. Furthermore, it can be seen that in traditional classifiers, KNN performs better. Moreover, the proposed algorithm predicts the maintenance configuration to store the large-scale IoT data with 95.87% accuracy.

We also perform the model complexity analysis of the proposed neural network model in terms of time complexity as shown in Figure 3.9d. The experiments are performed for different number of data instances, and model training and testing time is measured. It can be seen from the result that increasing the number of data instances results in an increased runtime of algorithm. The obtained results indicate that the proposed models execute in a reasonable amount of time for different instances of dataset. The obtained results show that the proposed approach is reasonably fast and its performance does not fluctuate much with the change in datasets.

Table 3.1 Pred-ms comparison

| Model | RMSE | MAPE |
|---|---|---|
| LSTM | 0.0699 | 0.0528 |
| Bidirectional LSTM | 0.0448 | 0.0349 |

We now evaluate the performance of a LSTM-based time series prediction model by using the LSTM module in the $keras$ library, with $tensorflow$ as a backend, to predict time series data.

**Parameters Settings:** We split our dataset randomly in the training and testing dataset. We also use min-max scaler in order to scale the data. Because of the existing temporal dependencies in the model, Back propagation (BP) method cannot be applied directly to the proposed model therefore, to train our LSTM-based model, we utilize back propagation through time (BPTT) model. The training data is split into several batches and processed sequentially. The weights

are updated upon completion of every batch. This process is repeated for all the defined batches. Moreover, we run the model several times independently to achieve the average performance metrics. The objective function is minimized by using the Adam optimizer.

We use two criteria to evaluate the performance of the proposed : Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). RMSE is calculated by squaring the difference between the actual and predicted values, while MAPE calculates the percentage of the absolute residual between the actual and predicted value. MAPE and RMSE are given by $\frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i-\hat{y}i|}{|yi|}$ and $\sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$ respectively.

In our proposed LSTM based model, we use LSTM as well as bidirectional LSTM in order to compare the performance of both models in terms of RMSE and MAPE. Table 3.1 shows the performance comparison of LSTM and bidirectional LSTM based model. It can be seen that bidirectional LSTM achieves better forecasting performance in terms of RMSE and MAPE, compared to LSTM-based model. Similarly, Table 3.2 shows the lower and upper bound of RMSE and MAPE values for LSTM and bidirectional LSTM network. The IoT data sample and service requirements time series data is not-stationary and non-periodic, and randomly changes with time. Using bidirectional LSTM, the learning algorithm is fed with the original training data once from beginning to the end and once from end to beginning. These two inputs in the bidirectional LSTM allow the model to capture the data trends better than single LSTM, where the model is fed with one input, i.e., from beginning to the end, thus leading to better performance.

Table 3.2 Lstm-based rnn model comparison

| Model | RMSE | | MAPE | |
|---|---|---|---|---|
| | Lower Bound | Upper Bound | Lower Bound | Upper Bound |
| LSTM | 0.0694 | 0.0702 | 0.0523 | 0.0530 |
| Bidirectional LSTM | 0.0439 | 0.0452 | 0.0342 | 0.0354 |

### 3.6.4 Blockchain cost analysis

In this subsection, we analyze the blockchain transaction and storage overhead. We employ Ethereum blockchain to implement the smart contract functionality of our proposed design however, any blockchain system with smart contract functionality can be used in our proposed design.

**Transaction Overhead:** We further investigate the transaction overhead of the proposed protocol over a period of 30 days in Figure 3.10a, in order to compare the total number of transactions with the baseline (our previous work in (Danish & et al., 2020)). Our proposed protocol incurs an average of 23 transactions per day, compared to an average of 12 transactions in the baseline solution. Different data generation pattern results in different number of transactions per day and transaction will only be recorded on the blockchain after the selection of new storage technology. The baseline solution incurs less transactions than the proposed blockchain-based solution however, the baseline solution doesn't provide traceability, auditability and decision verifiability to the storage technologies as well as IoT applications. Our proposed solution provides traceability, auditability and verifiability for the middleware design at the cost of extra transactions thus, the increased transaction overhead is justified.

We then perform a comparison in terms of transaction overhead per day, over a period of 30 days. We provide a comparison of two neural network models, i.e., binary classifier (baseline) and LSTM-based RNN network in Figure 3.10b. It can be seen that the proposed LSTM-based RNN network incurs less transaction overhead per day compared to the binary classifier baseline most of the days. Moreover, we can see that LSTM-based RNN network incurs almost 18 transaction per day on average compared to almost 23 on-average transactions of the baseline.

**Smart Contract Analysis:** We provide the gas cost estimation of the functions in Figure 3.10c. Gas cost is a unit that measures the amount of computational effort that smart contract will put to execute certain operations. In our work, the smart contract has four main functions: First, $publishIoT$ is used by the fog nodes to publish the IoT applications service requirements. Second, $publishSto$ is used by storage technologies to store their service parameters. Third,

*publishDec* is used by middleware to write decision in the blockchain network. Finally, with the help of *registration* function, the storage technologies register themselves in the middleware design. The *registration* function is executed once for every storage technology in order to register itself with the middleware. We provide the detailed gas cost estimation for these functions in Table 3.3.

Table 3.3 Gas cost analysis

| Operation | Gas usage | Safe Low | | | Fast | | |
|---|---|---|---|---|---|---|---|
| | | Gas price (in Gwei) | Gas cost (in ethers) | Confirmation time (in seconds) | Gas price (in Gwei) | Gas cost (in ethers) | Confirmation time (in seconds) |
| PublishIoT | 141546 | 80 | 0.0127 | 86 | 127 | 0.0179 | 36 |
| PublishSto | 162124 | 80 | 0.0145 | 86 | 127 | 0.020 | 36 |
| PublishDec | 141419 | 80 | 0.0127 | 86 | 127 | 0.0179 | 36 |
| Registration | 60900 | 80 | 0.005 | 86 | 127 | 0.007 | 36 |

## 3.7 Conclusion

In this work, we introduce BlockAIM, which jointly considers cloud and decentralized storage solutions. BlockAIM offers traceability, accountability, auditability and data integrity through a blockchain-based data placement protocol. In this protocol, we model the storage selection problem as a decision optimization problem and propose two heuristic algorithms: Dynamic Programming (DP)-based algorithm and Greedy Style (GS) algorithm. We then propose a neural network-based maintenance reconfiguration, which aims to optimize the computational complexity of the middleware design as well as the blockchain transaction and storage overhead by learning and predicting the associated parameters. We also propose aggregation rate feedback control for fog nodes and model it as a linear optimization problem, which optimizes for data quality and precision. Results show that the DP heuristic and GS heuristic achieves up to 92% and 80% accuracy respectively. Moreover, the proposed neural network-based binary classification model achieves accuracy up to 95.87%.

# CHAPTER 4

# BLOCKEV: A BLOCKCHAIN-BASED EFFICIENT CHARGING STATION SELECTION FOR ELECTRIC VEHICLES

Syed Muhammad Danish[1] , Kaiwen Zhang[1] , Hans-Arno Jacobsen[2] , Nouman Ashraf[3] , Hassaan Khaliq Qureshi[4]

[1] Département de génie logiciel et des TI, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Electrical & Computer Engineering, University of Toronto
[3] TSSG research group, Waterford Institute of Technology
[4] National University of Sciences and Technology (NUST), Pakistan

Paper published in *IEEE Transactions on Intelligent Transportation Systems*, November 2020

This core chapter provides the details of our blockchain-based CS selection in EV charging network. Section 4.1 provides an introduction and motivation for the considered research problem. Section 4.2 proposes a blockchain-based EV charging framework. Section 4.3 and 4.4 present the smart contract design and blockchain-based EV charging protocol, respectively. Section 4.5 formulates a CS selection decision optimization problem. Section 4.6 and 4.7 provides the security analysis and numerical results for the proposed protocol, respectively. The key conclusion are offered in section 4.8.

## 4.1 Introduction

Electric Vehicles (EVs) are emerging future Intelligent Transportation System (ITS) (Goli & Eskandarian, 2014; Zhou, Xiong, Xu, He & Mumtaz, 2017) application in smart cities (Blum & Eskandarian, 2007) and EVs have been recognized as a promising solution to reduce $CO_2$ emissions compared to traditional gasoline vehicles (Pajic, Rivera, Zhang & Jacobsen, 2018). The number of EVs are growing at a significant pace, and according to the national targets the availability of over 250 million EVs annually has been predicted annually by 2030 (Liu *et al.*, 2019). This dramatic growth will bring new challenges in managing the EV charging, therefore, to ensure long-term practicability of the EV industry, there is a need to manage EV charging efficiently to

provide vital comfort to the EV drivers. Previous works on EV charging management focus mainly on charging scheduling for EVs parked at charging stations (CSs) or homes to decide when EVs should be charged to optimize the efficiency of CSs (Rivera, Goebel & Jacobsen, 2016). Recently, with the increased deployment of public CSs with competing energy operators in a liberalized market, different CS providers offer dynamic pricing based on the electricity load and available charging level, i.e., slow or fast charging (Knirsch *et al.*, 2018). Also, peak pricing schemes might be in effect to shift load therefore, the research trend is now shifting towards which CS should be selected to charge a given EV (Cao *et al.*, 2019), as each energy provider offers different prices based on the current load, availability of energy and waiting time (Knirsch *et al.*, 2018). The International Renewable Energy Agency (IREA) considers smart charging and user incentives, like dynamic pricing, to be two key factors for unlocking the flexibility potential from EVs, which is required for a successful grid integration of EVs and renewable energy in the future. Moreover, the trend is currently moving towards dynamic pricing among competing charging networks in making EV public charging a viable business (Yu, Song, Su, Tang & Han, 2019), e.g., in United States, many charging station network operators (Electrify America, Tesla, Blink, ChargePoint, eVgo) are entering the market, which offer different charging prices per minute or per kilowatt hour (kWH) [1].

In the light of above mentioned dynamic electric charging network, in this paper, we address the first problem: which CS should be selected based on the service requirements of each charging EV while maintaining privacy for the EV's user. Specifically, we aim to transfer the decision making capabilities from the central management and the CSs to the EVs themselves in order to provide better privacy to the EV users. We consider a network of EVs and deployed CSs, where each travelling EV has varying service requirements in terms of price, charging time, travelling time and waiting time, and each EV is capable of selecting a CS in a distributed manner, based on its service requirements, while considering multiple CSs in its vicinity. Moreover, service requirements of some EV users are elastic, e.g., some users are willing to charge their EV for a lower price with more waiting and traveling time, while other EVs prefer short waiting time

[1] https://afdc.energy.gov/fuels/electricity_locations.html

but with a higher charging price. This CS selection mechanism by individual EVs will have important and long-term impact on the viability of a CS network, as it will alleviate congestion at certain CSs, i.e., CSs use pre-reservation for different time slots. Moreover, it will increase EV user comfort with an optimized price and better quality of service (QoS) to the EV users. Finally, the EV will be able to make a decentralized CS selection decision without sharing its private information to a CS or other network entities.

Currently, industry standard protocols are enabling communication between EVs and CSs in the charging network, e.g., ISO 15118 and Open Charge Point Protocol (OCPP) (Antoun *et al.*, 2020). However, these industry standard protocols suffer from severe security and privacy threats. The ISO 15118 protocol involves a trusted intermediary mobility operator, which keeps the private information (EV identity, location, state of charge, charging parameters, availability and payment information) of EVs to authenticate and identify them (Eiza *et al.*, 2018; Bao *et al.*, 2018). Moreover, it also tracks mobile EVs to route them to a suitable CS for charging, which raises serious concerns if the mobility operator leaks the EV's private information, intentionally or unintentionally (Bao *et al.*, 2018). Furthermore, this protocol offers no traceability and accountability mechanism in case the EV or the CS misbehaves in the charging protocol (Bao *et al.*, 2018). The OCPP charging protocol suffers from the same security and privacy issues (Antoun *et al.*, 2020; Khodari, Rawat, Asplund & Gurtov, 2019). In order to reserve a charging slot at a CS, the EV sends its private information to the trusted central management entity and then the central entity sends this information to a CS to reserve a charging slot for the EV (Antoun *et al.*, 2020). Furthermore, even in the presence of Transport Layer Security (TLS), OCPP is vulnerable to impersonation attacks where an attacker can pretend to be a CS or the central entity, to request or acquire the EV's private data regarding the charging transactions(Antoun *et al.*, 2020).

As the private information of the EVs is handled by a trusted centralized intermediary, the collection of such private data and tracking of an EV raise serious privacy and security concerns (Knirsch *et al.*, 2018; Au *et al.*, 2013). It has been identified that the private information of EVs has been used for position tracking of the vehicle as well as to predict the EV user's

habits, such as working hours, workplace and other important patterns (Knirsch *et al.*, 2018; Clarke & Wigan, 2011). Similarly, malicious businesses can perform targeted advertising against the EVs by using their location information (Knirsch *et al.*, 2018; Cao, Wang, Kamel & Kim, 2015). Furthermore, these centralized decision mechanisms are also vulnerable as they constitute a single point of failure. Finally, the centralized EV charging scheduling suffers from scalability issues, i.e., large number of EVs in the network leads to high communication and computational requirements (Nimalsiri *et al.*, 2019). Since, centralized EV charging protocols raise serious privacy and security concerns along with high communication cost, there is a need to remove trusted intermediaries and propose a decentralized EV charging protocol. Therefore, in the light of the above security and privacy concerns, we address the following problem: how the EVs can communicate with the CSs without sharing their private information, while keeping their identities secure.

Moreover, in the current EV charging protocol, the EV reserves a charging time slot at a CS by sharing its private information to trusted central management and the CS, which makes the reservation decision for the EV. Therefore, existing solutions require CS selection mechanism to be decided by the EV in a decentralized manner, without sharing the private information to other entities in the network. In addition to privacy concerns, the correct transaction logic of a charging reservation is often not properly implemented (Knirsch *et al.*, 2018). For instance, a CS can promise a certain price for a remote reservation of a specific time slot, which then changes when the EV eventually arrives. Conversely, an EV can make a remote reservation with a CS, but does not arrive on time. Thus, in the light of these trust issues among the EVs and the CSs, we address the following problem: how to enforce the EVs and the CSs to honestly honor the reservation once it has been made and make them accountable and auditable for their actions and misbehaviour's in the charging protocol (Bao *et al.*, 2018).

In order to address the above mentioned privacy and security problems, we propose the use of blockchain-based distributed ledger technology (DLT) (Pajic *et al.*, 2018; Li *et al.*, 2019). Existing works address the problems of EV charging protocols by employing blockchain-based solutions. However, they assume trusted third parties/central aggregators in their design, which

handle EV's private information thus, the privacy of EV has been largely ignored. Moreover, authors in (Knirsch *et al.*, 2018) propose privacy-preserving EV charging protocol however, the work lacks traceability, accountability and availability in case the EV or the CS misbehaves in charging protocol. Moreover, the work lacks theoretical formulation and technical details for privacy-preserving CS selection by the EV without disclosing its private information. Finally, the proposed design in (Knirsch *et al.*, 2018) incurs large transaction and storage overhead in the blockchain network. Hence, we address two main research challenges in this paper. The first research challenge is how an EV can make a reservation privately at a CS without sharing its private information to the CS or central entity with guaranteed availability of the CS, while enforcing interacting entities (EV and CS) to honour reservation and payment mechanism in an untrusted environment. The other research challenge is how an individual EV could make an efficient CS selection decision in a decentralized manner without leaking its private information to a CS and other network entities.

To address the first research challenge, we propose BlockEV, a blockchain-based EV charging protocol, which enables the EVs to communicate with the CSs without sharing their private information. It aims to ensure the trust, privacy, auditability and traceability and enforces EVs and CSs to honestly honor the energy trading protocol. Blockchains offer immutability, transparency, anonymity, availability and ensure trust in an untrusted environment without any trusted third party intermediary, which enables transparent, tamper-proof and secure energy trading between CSs and EVs, where an EV can make a reservation at a CS without sharing its private information to trusted intermediaries or CSs. The proposed BlockEV ensures the privacy of the EV users while interacting with CS to remotely reserve the charging slots at CS premises. Moreover, BlockEV also ensures the availability of the reserved charging slots at the CS, i.e., no charging reservation conflict among multiple EVs. Our proposed protocol also utilizes smart contract to enforce the charging reservation and payment logic between EVs and CSs to ensure honest behavior and to impose a penalty in case of deviation from normal behavior, while maintaining the privacy of the EV users with low transaction and storage overhead.

To address the second research challenge, in this work, we propose an efficient decentralized CS selection mechanism for the EVs, which allows EVs to select the CS themselves with high QoS and enhanced EV user comfort, while keeping EV's private information secure. We consider the decentralized charging framework in which CSs just broadcast their parameters such as price, charging slots, waiting time etc. and the decision is entirely made by EVs individually based on broadcast parameters. We theoretically formulate a decision optimization problem, which aims at minimizing the total cost to select the CS based on the service requirements of the EV. The paper makes the following key contributions:

- Unlike the previous centralized architectures, we propose a decentralized blockchain-based EV charging architecture, which eliminates the need of any trusted third party and enables the EVs and the CSs to communicate in a decentralized manner.
- We propose a blockchain-based EV charging protocol, which enables EVs to select a CS and make a remote reservation with a CS without the need of any trusted intermediary central management system. The proposed protocol enables EVs to reserve a charging slot at a CS remotely, thus preserving EV's privacy. Unlike previous works, the proposed protocol allows EVs to privately select a CS and remotely reserves a charging slot while ensuring the EV's privacy, security and guarantees availability of the CS (after reservation) at the same time. Moreover, the protocol also offers traceability, accountability and data integrity in case any entity in the network misbehaves.
- We propose the smart contract design, to enable the communication between the EVs and the CSs in an anonymous manner, in the untrusted environment. The smart contract guarantees the availability of the CS after a remote reservation, imposes penalty if the EV or the CS misbehaves in the charging protocol and defines reputation mechanism to rate a CS based on the EV user's experience.
- We analyze the detailed cost composition of EV's service requirements and theoretically model the decentralized CS selection mechanism as a decision optimization problem, where a EV selects the CS, while minimizing its service requirements in terms of price, charging time, waiting time and travelling time. Unlike previous works, EV itself is making a decentralized CS selection decision rather than sharing its information to central entity or CS.

- We provide security analysis and numerical results for the proposed protocol. We provide detailed security analysis of our protocol under various security attacks and provide detailed comparison with state-of-the-art schemes and industry standard protocols. In addition, we perform extensive simulations on real world dataset to demonstrate the effectiveness of the proposed blockchain-based framework by comparing with other conventional schemes.

## 4.2 Blockchain-enabled EV charging

In this section, we propose a blockchain-based EV charging framework for CS selection by individual EVs. Fig. 4.1 shows the proposed BlockEV system architecture with EVs, CSs, Road Side Units (RSUs) and blockchain network. Moreover, this section also explains the privacy requirement and threat model considered for the proposed protocol.

**Network Entities Definition:** The role of each entity in the architecture proposed in Fig. 4.1 is given below.

1. *Electric Vehicles:* The electric motors in the EVs are powered by rechargeable batteries. When the State of Charge (SoC) of EV reaches its threshold limit, EV makes a charging reservation with a CS based on its requirement. Each EV has a built-in smart meter (Saghezchi *et al.*, 2017) that records the traded energy amount in real time and is used to pay charging cost to CSs (Al-Rubaye, Rodriguez, Al-Dulaimi, Mumtaz & Rodrigues, 2019b).
2. *Charging Stations:* CSs are deployed geographically from different competitive operators throughout the city. Each CS has multiple charging slots i.e., it can charge multiple EVs in parallel. Moreover, CSs broadcast their charging information (price, waiting time, available charging slots, location and charging power) to RSUs through reliable communication channel.
3. *Road Side Units:* RSUs (Al-Rubaye, Al-Dulaimi & Ni, 2019a) are deployed on the sides of the road at fixed locations which disseminate the charging information of CSs to the EVs in their vicinity.

4. *Blockchain Network:* EVs and CSs performs reservation and payment through smart contract on the public blockchain network. Moreover, this blockchain network is also used as a storage layer to store reservation and payment information.

### 4.2.1 System model

We consider a network of $\mathcal{M}$ number of CSs from different competitive operators, deployed geographically in a city, where $m \in \mathcal{M}$ represents an arbitrary CS. These $\mathcal{M}$ CSs broadcast their charging information to $\mathcal{R}$ RSUs deployed geographically in a city, where $r \in \mathcal{R}$ represents an arbitrary RSU. A set of $\mathcal{G}$ EVs are operating in the city either in parking or on-the-move mode, where each EV has varying charging requirements, e.g., some EV users are willing to charge their EV for a lower price with more waiting and traveling time, while other EVs prefer short waiting time but with a higher charging price, where $g \in \mathcal{G}$ represents an arbitrary EV. These EVs receive the broadcast charging information of CSs through the $r^{th}$ RSU in their vicinity and solve a decision optimization problem to decide via which CS they need to recharge their batteries such that the total monetary as well time cost to reach and charge itself at selected CS, are jointly minimized. A public blockchain network is directly connected to the EVs and CSs, where each entity in $\mathcal{M}$ and $\mathcal{G}$ has a public account to interact with the smart contract and other entities in public blockchain network.

**Assumptions:** To broadcast the CSs information, two modes of communication can be used. RSUs can either use the cellular network or a Vehicular ad hoc network (VANET) to broadcast the information (Eskandarian, Chaoxian & Chuanyang, 2019; Yi *et al.*, 2019). However, in this work, we only consider VANET communication technology to disseminate the information from CSs to EVs through RSUs. A Publisher/Subscriber (P/S) push model can be used, where EVs as subscribers, passively receive the CSs broadcast information from a nearby RSU without any explicit query. EV will receive the CS broadcast information from RSU when the EV is within the radio coverage of that RSU. We assume a reliable communication channel in our model and as this is not the focus of this work, readers are referred to (Cao *et al.*, 2015) for detailed information of P/S VANET dissemination model. Moreover, each EV has a Global Positioning

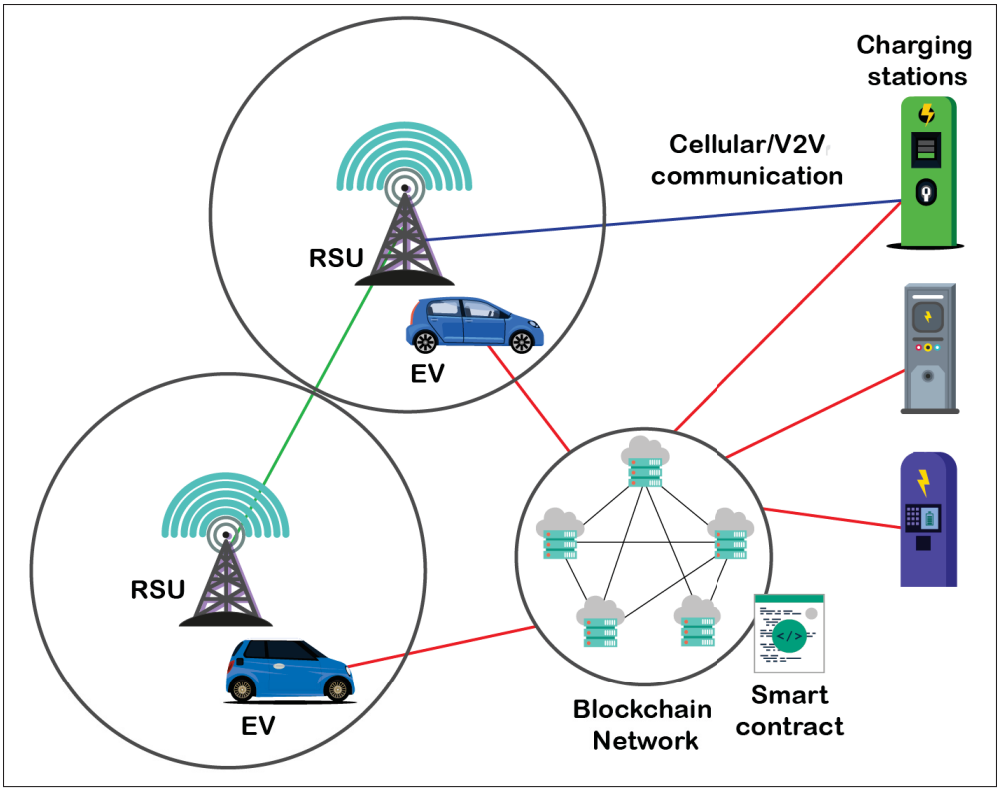



Figure 4.1 Blockchain-based EV architecture (Danish *et al.*, 2020b)

System (GPS) service, which provides information about speed and current location. Each EV in $\mathcal{G}$ also knows the location information of $\mathcal{R}$ RSUs and $\mathcal{M}$ CSs.

### 4.2.2 CS information broadcast scenario

In this work, we assume a smart city scenario where multiple CSs from different competitive operators are deployed geographically. Moreover, the CSs location information is publicly available for all the RSUs and EVs. RSUs are deployed across the city to handle control information between CSs and EVs. It is assumed that each CS is connected to all the RSUs through a reliable communication channel, e.g., authorized cellular network communication (Cao *et al.*, 2015). It takes the charging information from all the CSs connected to it and broadcast it periodically to all the EVs within its range. Each CS offers the charging service to the EVs in the available time slots with a certain charging price. Each individual EV takes a

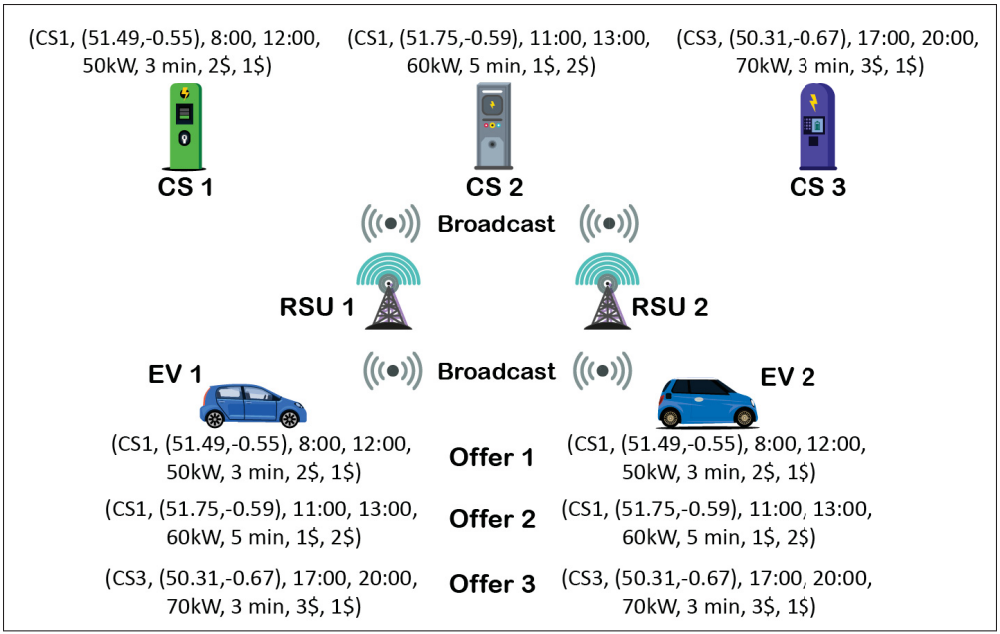



Figure 4.2 CS parameters broadcast scenario

decision in such a way that the selected CS fulfills all of its service requirements. The broadcast information of CS is defined as follows:

**Definition:** Each CS is characterized by < ID, loc, $T_s$, $T_e$, $\alpha$, $T_w$, $C_m$, $C_p$, $R_{CS}$ >, which defines the ID of CS, location of CS, start time of available charging slot, end time of available charging slot, charging power, estimated waiting time, charging cost per unit time, parking cost per unit time and reputation respectively.

Considering a charging time slot of one hour, the time between $T_s$ and $T_e$ is the available time period at CS to charge EV users and this time can have multiple time slots which can be selected by EV users based on their preference. Example. 1 explains a typical broadcast scenario of CS charging information to EVs through RSUs.

**Example 1.** Fig. 4.2 shows a typical broadcast scenario of charging parameters of different CSs. Three CSs broadcast their charging parameters including the available time slots and price to the connected RSUs through reliable communication channel for example, CS 1 broadcast the charging information characterized by < CS1, (51.49, -0.55), 8:00, 12:00, 50kW, 3 min, 2$,

1$ >. The RSUs then broadcast this information along with CS reputation, retrieved from the blockchain network to the EVs within their range characterized by < CS1, (51.49, -0.55), 8:00, 12:00, 50kW, 3 min, 2$, 1$, 7⋆ >. Now, as EVs have offers from different CSs, individual EV will use this information to solve the optimization problem derived in (4.21a) to select the CS which fulfills its requirements.

It must be noted that a CS can advertise different time slots at the same time. These offers can be considered simultaneously considering virtual CSs. For example, if CS 1 has two different charging slots available, it can be characterized by < CS1, (51.49, -0.55), 8:00, 9:00, 50kW, 3 min, 2$, 1$ > and < CS1, (51.49, -0.55), 10:00, 12:00, 50kW, 3 min, 2$, 1$ >.

### 4.2.3 Privacy and security requirements

The main objective of this protocol is to ensure the privacy as well as trust environment between EVs and CSs, where an EV can select the best suited CS based on its requirements among all the possible CSs within the range. These CSs include the vacant private and public CSs at the time of reservation. However, none of the involved parties should learn the customer's position and energy need during this phase. In order to preserve the private information of the EV users while selecting the CS to charge itself, the following requirements needs to be fulfilled in the protocol.

- No entity taking part in the protocol gets information about the EV's exact position.
- No entity taking part in the protocol gets information about the EV's current SoC and required energy.
- No participant in the protocol gets information about the purchased energy price except selected CS.
- No entity in the protocol gets information to track EV vehicle over time i.e., after leaving the CS premises, CS gets no information about EV.
- No participant in the protocol learns the decision parameters of corresponding EV except EV itself.

Except these privacy requirements, another objective of this protocol is to ensure the trust environment among the protocol participants i.e., the CSs or EVs should follow the normal protocol and deviating from normal behavior incurs a monetary as well as reputation penalty.

**Threat Model:** In this work, we consider malicious participants in the EV charging market which may threaten EV's privacy and security. We assume that the government has deployed the RSUs geographically in the city and are operating normally and honestly. We consider RSUs to be an honest entity because the government defines the security specification for the deployment of RSUs. For example in United States, the security requirement for the deployment of RSUs is defined in the official document by the US department of Transportation [2]. As these RSUs are deployed under United States department of Transportation vigilance, we assume that these RSUs are honest and government controls physical access to these RSUs. However, we define malicious adversaries as:

- *Malicious Charging Station:* A malicious CS can broadcast fraudulent charging parameters without enough time slots availability, charging power $\alpha$ etc. Moreover, some CSs may not fulfill the charging requirements of EVs, made at the time of reservation. They can also try to use different charging parameters at the time of reservation and try to make a reservation with different parameters.
- *Malicious Electric Vehicle:* A malicious EV may agree for charging services from selected CS and later refuse to pay the incurred charging cost to CS. Moreover, EV can also reserve the charging slot with a certain CS and never appears at the reserved slots.

Furthermore, it is assumed that all the cryptographic primitives and security protocols are secure and cannot be compromised.

### 4.3 Smart contract design

In this section, we present the smart contract design of the proposed BlockEV protocol. Smart contracts are deployed within the blockchain network with digital commitments among the

[2] https://rosap.ntl.bts.gov/view/dot/3600

protocol participants. Smart contract is a piece of code which is executed automatically when a node triggers it by sending transaction without the involvement of third parties (Zhang & et al, 2018). In BlockEV smart contract enables secure and verifiable communication between EVs and CSs to enforce honest reservation and payment mechanism in an untrusted environment. The four main smart contract functions associated with the proposed protocol are $reservation$, $authentication$, $payment$ and $reputation$ as shown in Algorithm 4.1. It has been assumed that the EVs and CSs will be able to exchange money in terms of cryptocurrencies.

### 4.3.1 Reservation

When the EV has selected the CS based on its service requirements (*step 3*), the EV needs to make a remote reservation with the selected CS. This reservation is made through the smart contract in the blockchain network to make sure:

- CS is available for the charging request at reserved time and does not change the price for the charging request after arrival of EV at the CS.
- EV should not make a fake reservation at the CS for the selected charging time slots.

The $reservation$ function will take input from the EV and CS and compare the information before sending as a transaction to blockchain network to make sure the EVs gets the same cost parameters which were used to select the CS in (*step 3*). The EV and CS will send the following information as an input to *reservation* function.

$$\text{EV, CS} \rightarrow SC_{reservation}\colon \left\{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_s||deposit\right\}_{EV} \cup \left\{C_m||C_p||T_s||(randomnumber)_{pk_{EV}}\right\}_{CS}.$$

The $reservation$ function will make sure that the EV has enough coins to pay for the deposit as well as charging cost. Moreover, the function will also compare the charging price and time slot information from EV and CS as follow.

$$SC_{reservation}: compare = \left\{C_{m_{EV}}||C_{m_{CS}}, C_{p_{EV}}||C_{p_{CS}}, T_{s_{EV}}||T_{s_{CS}})\right\}.$$

Algorithm 4.1 Smart contract design

**Reservation:**
Input: $T_A$, $SoC_{curr}$, $SoC_{des}$ , $C_m$, $C_p$, $T_s$, $deposit$, $(RN)_{pk_{EV}}$
Compare($C_{m_{EV}}||C_{m_{CS}}, C_{p_{EV}}||C_{p_{CS}}, T_{s_{EV}}||T_{s_{CS}}$)
**if** $C_{m_{EV}} = C_{m_{CS}}, C_{p_{EV}} = C_{p_{CS}}, T_{s_{EV}} = T_{s_{CS}}$ **then**
    **if** $deposit_{EV} \geq threshold$ **then**
        *make reservation and execute on blockchain*
    **end if**
**end if**
**Authentication:**
Input: $dec_{sk_{EV}}((RN)_{pk_{EV}}, (RN)_{CS}$
**if** $dec_{sk_{EV}}((RN)_{pk_{EV}} == (RN)_{CS}$ **then**
    *Authenticate EV*
**end if**
**Payment:** Input: $ID, T_{total}, C_m, C_p, SoC_{des}, SoC_{curr}, deposit_{init}$
**if** *charging is ended* **then**
    $transfer_{EV \rightarrow CS}$ $(T_{total}(C_{m_{CS}} + C_{p_{CS}}) - dep_{init})$.
**end if**
**if** $SoC_{char} \leq SoC_{des}$ **then**
    $deposit_{CS \rightarrow EV}(Cost^{EV}_{pen})$.
**end if**
**if** $T_{sa} \geq T_{aa}$ **then**
    $deposit_{EV \rightarrow CS}(Cost^{CS}_{pen})$.
**end if**
**Reputation:**
**if** *charging is completed* **then**
    *rate CS.*
**end if**

If the required information matches, then the function will execute and a transaction will be sent to the blockchain network thus, reservation will be written on the immutable distributed ledger. It should be noted that if the deposit functionality is implemented without the blockchain network to prevent malicious EVs to make fake reservation, the privacy of EV would not be preserved as the EV can be linked with its credit card information either by trusted third party or CSs.

**Reservation Significance:** As the electricity price is dynamic and depends on the current load, it is necessary to agree on a price during the reservation, so that even at the EV's reserved charging time slot if the electricity price changes, EV should pay the same amount as agreed in the remote reservation. Smart contract ensures that once the price is fixed at reservation, neither CS nor EV can change it later, thus ensuring trust in untrusted environment.

### 4.3.2 Authentication

Once the reservation has been made between EV and CS, EV will reach the CS premises on the reserved time. CS needs to authenticate the EV before starting the vehicle charging to make sure it is the same vehicle who made the remote reservation. EV already has a random number at this point so it will decrypt the random number through its $sk_{EV}$ and sends as an input to the authentication function. The authentication function will then retrieve the random number from the blockchain network which was added in the previous transaction of reservation function. The $authentication$ function will then compare the two random numbers as follows.

$$SC_{auth} : compare =$$
$$\{dec_{sk_{EV}}((randomnumber)_{pk_{EV}})||(randomnumber)_{pk_{EV}}\}.$$

If the random numbers match, the CS will authenticate and start charging the EV for available time slot.

### 4.3.3 Payment

The $payment$ function handles the payment between the EVs and CSs once the charging of the EV has been finished at the charging station based on the time taken to reach $SoC_{des}$. The function will take the initial time to charge at the start and automatically reads the charging end time and $SoC_{curr}$ from the smart meter of EV and CS in order to verify if there is an actual exchange of mentioned amount of transferred energy. Afterwards, the $payment$ function will

transfer the total cost from EV's account to CS's account with deduction of deposit amount EV made during the reservation as follows.

$$EV \rightarrow SC_{payment} \rightarrow CS\text{: } transfer = \left\{T_{total}(C_{m_{CS}} + C_{p_{CS}}) - deposit_{init}\right\}.$$

Moreover, the $penalty$ function is responsible for calculating the penalty for EVs and CSs in case of not fulfilling the conditions set in the reservation function. If EV does not arrive with in some threshold time, the CS is going to charge some penalty $Cost_{pen}^{CS}$ as given in Eqn. (4.1) and (4.2),

$$d = t_{aa} - t_{sa} \leq t_{threshold}, \tag{4.1}$$

$$Cost_{pen}^{CS} = P_t \times d. \tag{4.2}$$

where $P_t$ is some penalty factor, while $t_{sa}$ and $t_{aa}$ are the reserved arrival time and actual arrival time respectively. Similarly, if the CS doesn't fulfill the desired SoC requirement of EV (i.e., $SoC_l = SoC_{des}$ - $SoC_{char}$), EV needs to impose some penalty $Cost_{pen}^{EV}$ on CS as given in Eqn. (4.3) and (4.4),

$$SoC_l = SoC_{des} - SoC_{char}, \tag{4.3}$$

$$Cost_{pen}^{EV} = P_{SoC} \times SoC_l. \tag{4.4}$$

where $P_{SoC}$ is some penalty factor for $SoC$. This function will be called automatically after the payment function execution and it will take the $SoC_{des}$ and $t_{aa}$ from the blockchain network, while $SoC_{char}$ value will be retrieved from EV's smart meter, to calculate penalty amounts.

**Payment Significance:** The payment mechanism can be implemented in our design simply without blockchain however, in the current industry standards EV charging protocol, false pricing information can be injected in the system, which affects the payment mechanism (Rubio, Alcaraz & Lopez, 2018). Moreover, through EV impersonation, an attacker can use EV's private information for public charging billing purpose (Rubio *et al.*, 2018). Therefore, we implement the payment functionality in blockchain to make sure no false information can be injected in the system and EV should securely pay only for the provided service.

### 4.3.4 Reputation

The EV user will rate the CS based on its user experience through *reputation* function. The reputation $R$ of CS at certain charging request $t$ will be either positive ($R_t$ = +) or negative ($R_t$ = -) based on fulfillment of $SoC_{des}$, the actual charging power $\alpha$ at CS premises and the availability of reserved time slot $T_A$ as agreed in reservation.

$$EV \rightarrow SC_{payment}\colon transaction = \{Reputation_{CS_m}\}.$$

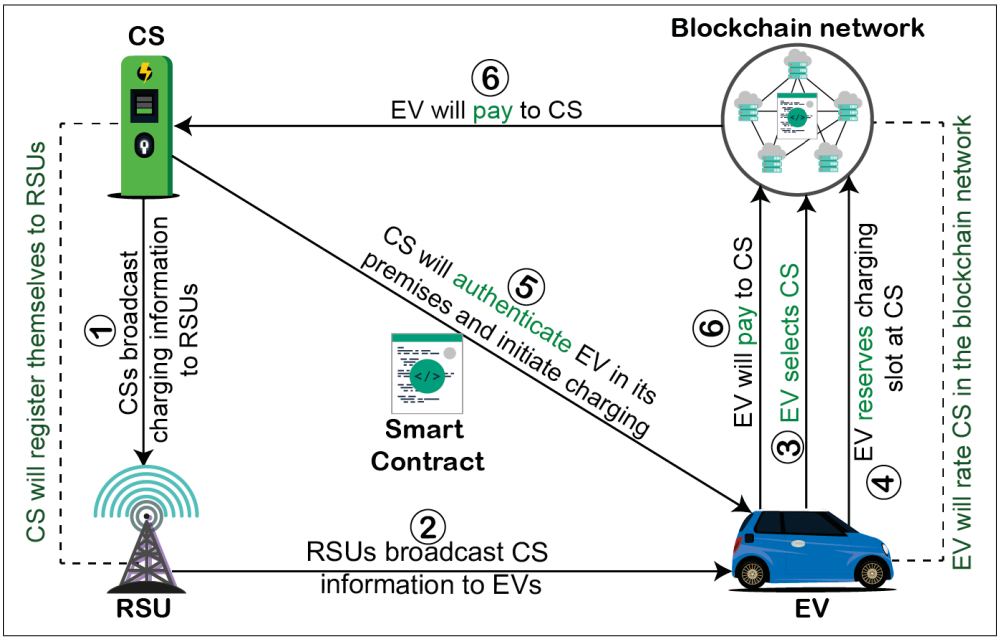



Figure 4.3 Ev charging protocol

## 4.4 Blockchain-based protocol

In this section, we propose BlockEV, a complete blockchain-based EV charging protocol, which includes EV's CS selection decision as well as its interaction with the selected CS on the blockchain network with smart contract functionality including reservation, authentication, payment and reputation. Furthermore, this section also explains the CS reputation mechanism.

Algorithm 4.2 EV charging protocol

Initialize $list_{CS} \leftarrow 0, m \leftarrow 0$
**while** *true* **do**
  $CS_m \rightarrow$ RSU : $\{ID_{CS_m}||pk_{CS_m}||Cert_{CS_m}\}$
  **if** $Cert_{CS_m} = valid$ **then**
    $list_{CS}.add(CS_m)$
  **end if**
  i = i + 1
**end while**
**Blockchain-Based Protocol:**
Initialize $ID, loc, T_s, T_e, \alpha, T_w, C_m, C_p, T_s, RN, t$
$ID, loc, T_s, T_e, \alpha, T_w, C_m, C_p, T_s, RN \leftarrow 0$
$t \leftarrow 1$
**while** *true* **do**
  *step 1:* CSs broadcast charging information to RSUs
  $CS_m \rightarrow RSU$ : $\{ID_{CS_m}||loc_{CS_m}||T_s||T_e||\alpha||T_w||C_{m_CS_m}||C_{p_CS_m}\}$
  *step 2:* RSUs retrieve $Rep$ from blockchain network
  $RSU \leftarrow Blockchain : request_i = \{(R_{CS_m})_{list}\}$
  $Rep_{CS_m} = \frac{\sum_{i=1}^{N} R_{t_i}}{N}$
  $RSUs \rightarrow EVs$ : $\{ID_{CS_m}||loc_{CS_m}||T_s||T_e||\alpha||T_w||$
  $C_{m_CS_m}||C_{p_CS_m}, Rep_{CS_m}\}$
  *step 3:* EV selects CS based on its requirements.
  EV solves optimization problem (4.21a) s.t. (3.7b) - (4.21h).
  *step 4:* EV makes reservation with selected CS.
  $SC_{res} \rightarrow Blockchain$ : $\{T_A||SoC_{curr}||SoC_{des}||C_m||$
  $C_p\| T_s \|$ deposit$\}_{EV} \bigcup \{C_m||C_p||T_s||(RN)_{pk_{EV}}\}_{CS}$.
  *step 5:* CS authenticates EV.
  $SC_{auth}$ : $match\ (dec_{sk_{EV}}((RN)_{pk_{EV}}), RN)$.
  *step 6:* EV pays CS after charging.
  $EV \rightarrow CS_m$ : $payment = \{T_{total}(C_{m_CS_m} + C_{p_CS_m})$
  t = t + 1
**end while**

### 4.4.1 EV charging protocol

Although blockchain-based bidding EV charging protocol has been proposed in the literature (Knirsch *et al.*, 2018), it does not always ensure the availability of reserved time slot at selected CS and can result in time slot conflict between two EVs. Also, it incurs high storage and

transaction cost in the blockchain network which impacts the profitability of the CSs taking part in the protocol. We propose BlockEV to address these problems.

Fig. 4.3 presents the protocol interaction among different entities in EV charging network. A public blockchain network is employed to support the smart contract functionality for the interaction between EVs and CSs. Moreover, the blockchain network is also used as a storage layer to provide auditability and traceability of reservation and payment information. At the basic level, the blockchain-based EV charging algorithm will work as in Algorithm. 4.2. More detail about blockchain-based CS selection protocol is given below:

**CS registration:** All of the CSs will first generate a set private ($sk_{CS_m}$) and public key ($pk_{CS_m}$). They will register themselves to the RSUs deployed geographically by providing their $ID$, public key and digital certificate ($Cert_{CS_m}$) to the RSUs, namely,

$$CS_m \rightarrow RSUs : request_{CS_m} = \left\{ID_{CS_m}||pk_{CS_m}||Cert_{CS_m}\right\}$$

Once the request has been received, RSUs will check the validity of the $Cert_{CS_m}$ for $m^{th}$ CS and add it in the list of available CSs.

**Blockchain-Based Protocol:**

- *Step 1:* All of the CSs periodically publish their charging information to the RSUs connected to it at current time $t$. This charging information includes the CS ID, charging price, location, start and end time of available charging slots, charging power and expected waiting time, namely,

  $$CS_m \rightarrow RSUs : request_i = \left\{ID_{CS_m}||loc_{CS_m}||T_s||T_e||\alpha||T_w||C_{m_{CS_m}}||C_{p_{CS_m}}\right\}.$$

  Every charging information at time $t+1$ will replace the charging information at $t$. Therefore, the RSU will replace the charging information of $CS_m$ at time $t+1$ by newly arrived charging information at $CS_m$.
- *Step 2:* Once the RSUs receive the charging information from the CSs connected to them, they will retrieve the reputation information (rating) of the CSs from the blockchain network, namely,

$$RSU \rightarrow Blockchain : request_i = \{Rep_{CS_m}\}.$$

where $Rep_{CS_m}$ is the reputation of $m^{th}$ CS. The blockchain network will reply with reputation value corresponding to the requested CS $ID$, namely,

$$Blockchain \rightarrow RSU : response_i = \{Rep_{CS_m}\}.$$

After retrieving the information from the blockchain network, the RSUs will add corresponding reputation value to CS charging request and broadcast it to the EVs within its range, namely,

$$RSUs \rightarrow EVs : request_i = \{ID_{CS_m}||loc_{CS_m}||T_s||T_e||\alpha||T_w||C_{m_CS_m}||C_{p_CS_m}||Rep_{CS_m}\}.$$

- *Step 3:* After receiving the broadcast CS charging information from the RSUs, the individual EVs who are willing to charge their vehicle will make the CS selection by solving an optimization problem based on their individual service requirements in terms of < *charging price*, *traveling time*, *waiting time*, *charging time*, *reputation* >. EV will calculate the cost parameters associated with the $m^{th}$ CS by,

$$ChargingPrice = C_{m_CS_m} + C_{p_CS_m}, \tag{4.5}$$

$$travelingTime = \min_{\forall i \in N} d(EV, CS_m), \tag{4.6}$$

$$ChargingTime = \frac{(SoC^{des} - SoC^{cur})}{\alpha_{CS_m}}. \tag{4.7}$$

where d is the distance function, while $SoC^{des}$ and $SoC^{cur}$ are the desired and current state of charge of EV respectively. EV will then use these parameters to solve a decision optimization problem to find the best-suited CS. We formally formulate the EV's efficient CS selection optimization problem in Section. 4.5. It should be noted that the EV has not shared any information with the CSs or RSUs and therefore its private information, e.g., location, current energy etc. is secure.

- *Step 4:* Once an individual EVs has made a decision about CS selection based on its requirement, it will reserve the charging slot with the selected CS through smart contract

*reservation* function. The EV will send the information as an input to *reservation* function as follows.

$$EV \rightarrow SC_{reservation} : input = \{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_{sel}||deposit_{init}\}.$$

where $T_A$ is expected arrival time, while $T_{sel}$ is the selected available time slot. $SoC_{curr}$ and $SoC_{des}$ are current and desired SoC at the end of charging, whereas $deposit_{init}$ is the amount EV needs to deposit at the time of reservation. Similarly, the CS will send the information as an input to *reservation* function as follows.

$$CS \rightarrow SC_{reservation} : input = \{C_m||C_p||T_{sel}||(randomnumber)_{pk_{EV}}\}.$$

where *random number* will be generated by CS to authenticate the EV at its premises and will be encrypted by the public key of EV i.e., $pk_{EV}$. Once the reservation function is called, a transaction will be sent to the blockchain network as,

$$SC_{reservation} \rightarrow Blockchain : \{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_{sel}||deposit\}_{EV} \cup \{C_m||C_p||T_{sel}||(randomnumber)_{pk_{EV}}\}_{CS}.$$

It should be noted that even at the time of reservation, CS does not know about the identity of EV. The identity of EV will only be revealed after reaching the CS premises. In the auction-based reservation mechanism in (Knirsch *et al.*, 2018), the CS is not sure if it has won the bid and is selected by the EV. This motivate CSs to bid multiple times for the same time slot thus, leading to availability issue for more than one EV. However, in our work, once the reservation has been made, no time slot availability conflict will occur because in the normal protocol settings, when an EV will make a remote reservation with CS, the smart contract will assure CS that reservation has been made and thus, CS will not be allowed to reserve the same time slot for another EV after making reservation with actual EV user.

**Conflicting Reservation:** Even if the CS is allotting the same time slot for more than one EV, the EVs will get their deposit back directly from the smart contract in case of acquiring no charging service from CS and they can provide a bad review to CS, which will decrease its probability to be selected by EV next time.

- *Step 5:* Once the reservation has been made by EV it will reach the selected CSs within predefined time and authenticate itself to CS through smart contract's *authentication* function to make sure it is the same EV who made the reservation. The EV will decrypt the random number with its private key and send the information as an input to *authentication* function as follows.

$$EV \rightarrow SC_{authentication} : input = \left\{dec_{sk_{EV}}((randomnumber)_{pk_{EV}})\right\}.$$

where $dec_{sk_{EV}}$ represents decryption function with the private key of EV. The CS will send the original random number as an input to *authentication* function as follows.

$$CS \rightarrow SC_{authentication} : input = \{randomnumber\}.$$

Based on this information, the smart contract will automatically authenticate the EV at CS premises. It should be noted that the real identity of EV will be released at this step i.e., after reaching the CS premises. Thus, the private information of EV is not released to any entity before reaching the selected CS premises. Moreover, information is also not revealed to other CSs, which were being considered during selection process but were not selected.

In this work, we only consider privacy from EV's point of view as CS is broadcasting its parameter publicly. Sharing the location of CS will leak the private information related to that certain CS even if we hide its identity therefore, we consider it a potential open problem i.e., how to provide the privacy to EV and CS at the same time, while ensuring CS availability.

- *Step 6:* After authenticating the EV at the premises, the CS will start charging the vehicle. It has been assumed that each EV has smart meter which takes real-time value of charged energy. The starting and ending time of the charging will be sent as an input to the *payment* function of the smart contract as follows.

$$EV \rightarrow SC_{payment} \rightarrow CS: input = \{T_{start}, T_{end}\}.$$

Based on the payment and reservation information, the smart contract will calculate the total price of charging as well as any monetary penalty imposed on the EV or CS. Once the EV has paid for the charging service to CS, it will leave the CS premises and enter in the driving phase again.

**Reputation:** The final step of this protocol is to define the reputation mechanism through which EVs can rate the CSs based on their experience with CS. It has been assumed that an honest EV will always provide an honest reputation review to CS for its services. In this protocol, we only emphasize on the reputation review of CSs and not the EVs because our main aim is to provide privacy to the EV users. If the reputation reviews against EV IDs can be seen by everyone on the blockchain network, the privacy of EV will not be preserved. Moreover, reputation review for EV is not possible as BlockEV does not share the ID of EV with any entity in the network and without sharing the EV ID, reputation is not possible. Therefore, we propose a reputation mechanism for CS in order to enforce the good behavior of CS in favor of EVs. The reputation $R$ of CS at certain charging request $t$ will be either positive ($R_t$ = +) or negative (positive ($R_t$ = -)) based on fulfillment of $SoC_{des}$, the actual charging power $\alpha$ at the CS premises and the availability of reserved time slot $T_A$ as agreed in reservation. The reputation value (between 1 to 10) of certain CS will be sent as an input to the *payment* function of the smart contract as follows.

$$EV \rightarrow SC_{payment}\colon input = \left\{Reputation_{CS_m}\right\}.$$

Therefore, in the step 2 of the protocol, the RSU will retrieve all the reputation information against the CS $ID$ and calculate the *average reputation value* ($TR_{CS_m}$) as,

$$TR_{CS_m} = \frac{\sum_{i=1}^{N} R_{t_i}}{N} \tag{4.8}$$

where N is total number of reviews $CS_m$ got from the EVs.

It must be noted that the EV is assumed to be honest in the protocol participation and we assume that EV will always rate the CS in an honest manner. Considering the malicious or semi-honest EV scenario, a wrong reputation value can be written in the blockchain network. This problem can be solved in future by using the decentralized storage technologies and smart contract with admin rights, where the reputation value can be changed or overwritten.

**Benefits of blockchain:** In this work, introducing blockchain network will remove the need of trusted intermediary central management, which will enable EV to make reservation at CS

without sharing any private information. Moreover, as no entity in the network is keeping EV's private information, EV cannot be tracked with time. Finally, as the reservation and payment data will be stored on the blockchain network, if any party tries to misbehave, this deviation from normal behavior can be traced back and the malicious party can be held accountable for their action thus, ensuring traceability, accountability and data integrity.

### 4.4.2 Economic perspective

The global economy is increasingly competitive, with growing demands that cannot be sustainably met by conventional energy systems. Thus, in order to meet the current economy demands, the trend is now moving towards a peer to peer (P2P) electricity trading, where the traditional energy consumers are becoming prosumers, who can both consume and generate energy. Blockchain-based P2P energy trading has been emerged as a possible solution to the challenges in the energy sector, asserted by energy sector decision makers and utility companies (Andoni *et al.*, 2019). The energy market operations will be more efficient and transparent with the help of blockchain (Andoni *et al.*, 2019). The most popular use case of blockchain-based power grids today is the Brooklyn Micro-grid (Andoni *et al.*, 2019). It is one of the first projects, leveraging blockchain technology for P2P energy trading, developed by LO3 Energy Inc. The aim of this project is to study how the blockchain technology may enable the instant trading of solar energy between neighborhoods.

Blockchain-based EVs charging network is also an example of P2P energy trading between CSs and EVs. Blockchain technology enables automated buying, selling, and scheduling of transactions among EVs and CSs with the help of smart contracts, based on their preferences. Moreover, real-time demand and supply in the EV charging network via smart contracts will automate the encrypted sale and purchase, thus leading to a secure business model (Andoni *et al.*, 2019). The proposed blockchain-based EV charging network allows EVs to select the charging station and schedule their charging slot without the need of any third party. Therefore, EVs are communicating and charging themselves in a competitive P2P market with CSs, where the CSs are offering competitive prices for the charging services to EVs. Unlike (Knirsch *et al.*,

2018) where all the CSs need to participate in the bid by writing the offered services on the blockchain network, the proposed model writes the transaction on the blockchain network once the deal is confirmed among EVs and CSs, thus leading to cost-effective solution for both EVs and CSs. From the business perspective, EVs will be able to make the independent charging decision based on the best price and other service requirements. From the CSs perspective, the deposit amount from EVs at the time of reservation will ensure the profit of CSs even if the EVs deviates from the normal behavior. Therefore, the proposed model enables the a competitive business model, which allows CSs to take part in the market by offering their services and allows EVs to select the CSs, best suited for their service requirements.

## 4.5 Charging station selection

In this section, we narrow down our focus on the theoretical formulation of CS selection problem in the proposed BlockEV. We formulate a decision optimization problem to select the CS based on EV's service requirements, while minimizing the overall cost associated with the CS in terms of price, travelling time, waiting time and charging time. Selecting a CS in a random manner is not a good solution to employ because it can result in the selection of CS with high price, more travelling time, waiting time or charging time etc. In this paper, we have analyze the detailed cost composition of EV's service requirements and use this cost model to devise a decision optimization problem, where the aim is to minimize all these cost to increase the QoS with enhanced user comfort. Thus, an EV is solving the decision optimization problem efficiently to select the best-suited CS for charging itself by minimizing all the overall cost associated with the service requirements.

### 4.5.1 Problem formulation

We consider the problem of how to minimize the overall cost of an charging an EV by efficiently selecting the CS, while keeping the privacy of EV user. The RSUs receive the service information from the CSs connected to it and broadcast it to the EVs in its range. This information is then used by the individual EV to decide the optimal CS. An EV user needs to decide via which

CS it needs to recharge its batteries such that the total monetary as well time cost to reach and charge itself at selected CS, are jointly minimized. In this work, we model the EV's decision for selecting an optimal CS as a decision optimization problem in which an EV user needs to decide a CS based on its service requirements. The total cost of the requirements that needs to be minimized has four components: monetary cost, waiting time cost, traveling time cost, charging time cost.

**Decision Variable:** Let $\mathcal{M}$ be the total number of CSs that are offering the charging service to the EV users. A binary decision variable $y_m$ is considered which corresponds to the selection of $m^{th}$ CS for an individual EV user. We require that for every decision model solution, a single CS should be selected through solving the optimization problem by EV user. Let $\overline{y} = (y_m)_{\forall m \in \mathcal{M}}$, the set of all possible decision variables is given by Equation. (4.9):

$$\mathcal{Y} = \{\, \overline{y} \mid \sum_{m \in \mathcal{M}} y_m = 1 \text{ and } y_m \in \{0, 1\}, \forall m \in \mathcal{M} \,\} \tag{4.9}$$

**Overall Cost:** The overall cost of an EV user, which needs to be jointly minimized is given by:

(1) The ***charging monetary cost*** refers to the price EVs need to pay for parking as well as to charge their vehicle at the charging station. Let $C_m \in \mathbb{R}^+$ and $C_p \in \mathbb{R}^+$ denote the EV's per unit time electricity charging cost and parking cost respectively at $m^{th}$ CS. Moreover, let $\mathcal{M}$ = $\{1, 2, 3, ..., M\}$ be the total number of CSs near an individual EV, offering charging service. The EV should choose the CS with minimum charging price as given in Equation. (4.10),

$$C_{price}(EV, CS_{sel}) = \min_{\forall m \in \mathcal{M}} (C_m(EV, CS_m) + C_p(EV, CS_m)) \tag{4.10}$$

where $C_{price}$ is the total price of selected CS, while $CS_{sel}$ is the selected CS. Thus, the total monetary cost to charge an EV is given by Equation. (4.11),

$$C_{mon}(\overline{y}) = y_m \mathcal{T}_{ch}(C_m + C_p) \tag{4.11}$$

where $\mathcal{T}_{ch}$ is the total time required to charge an EV. In this work, we only consider fast charging scenario at the CS premises therefore, we omit the parking time once the EV has been charged to its desired state.

(2) The ***traveling time cost*** refers to the cost incurred while reaching the $m^{th}$ CS. Let $\mathcal{M} = \{1, 2, 3, ..., M\}$ be the total number of CSs around an individual EV. The EV should choose the CS with minimum distance as given in Equation. (4.12),

$$d_m(EV, CS_{sel}) = \min_{\forall m \in \mathcal{M}} d(EV, CS_m) \tag{4.12}$$

Moreover, the travel time between current location of EV and CS is a function of distance $d_m$, and other factors, e.g., weather, speed, traffic conditions etc. The travel time can be calculated by Equation. (4.13),

$$T_{trip} = f(d_m, \mathcal{F}) \tag{4.13}$$

where $\mathcal{F}$ denotes the factors including speed of EV, weather and traffic conditions etc and can be represented as Equation. (4.14) (Chen, Zhang, Pourbabak, Kavousi-Fard & Su, 2016),

$$\mathcal{F} = \tau_{ij}(1 + \delta(\frac{V}{C})^{\theta}) \tag{4.14}$$

where $\tau_{ij}$ represents free flow time, V for traffic flow and C for road capacity. $\delta$ and $\theta$ are experimental coefficients and its value can be taken from (Chen *et al.*, 2016). Thus, the total cost of traveling can be represented by Equation. (4.15),

$$C_{tra}(\overline{y}) = y_m W_T T_{trip} \tag{4.15}$$

where $W_T$ represents the weight to convert the traveling time cost to monetary cost. This traveling time ($T_{trip}$) information can also be retrieved by google maps[3] as it provides multiple paths to a single destination. A path with minimum traveling time can be chosen.

[3] https://www.google.com/maps

(3) The ***charging time cost*** refers to the charging time for an EV to reach its required battery capacity. The total time to charge an EV vehicle is given by Equation. (4.16),

$$T_{tot} = \frac{(SoC^{des} - SoC^{cur}) + E_{tra}}{\alpha^m} \tag{4.16}$$

where $SoC^{max}$ is the desired state of charge, while $SoC^{cur}$ is the current state of charge of the EV's battery. $\alpha_m$ is the charging power at the $m^{th}$ CS. $E_{tra}$ denotes the electricity amount consumed during travel to the CS and is represented by Equation. (4.17),

$$E_{tra} = d_m \beta \tag{4.17}$$

where $d_m$ is the distance between EV and $m^{th}$ CS. $\beta$ denotes consumption of energy per meter and is specific to EV model. Thus the total cost of EV charging time is given by Eqn. (4.18),

$$C_{char}(\overline{y}) = y_m W_C T_{tot} \tag{4.18}$$

where $W_C$ represents the weight to convert the charging time cost to monetary cost.

(4) The ***waiting time cost*** refers to the time an EV needs to wait before starting the battery charging. Normally, a CS has multiple ports where the EVs are charged in parallel. Thus, the total cost of EV waiting time is given by Equation. (4.19),

$$C_{wait}(\overline{y}) = y_m W_W T_{wait} \tag{4.19}$$

where $W_W$ represents the weight to convert the waiting time cost to monetary cost, while $T_{wait}$ is the total time EV needs to wait at the selected CS. The waiting time at the CS is normally forecasted with the help of machine learning techniques. In this work, we assume that the CS has already forecasted this parameter using the past data and provided this information to the EV in the broadcast design.

**Optimization Problem:** The objective functions that need to be minimized can be represented as Equation. (4.20),

$$\mathbb{C}(\overline{y}) = C_{mon}(\overline{y}) + C_{tra}(\overline{y}) + C_{char}(\overline{y}) + C_{wait}(\overline{y}) \tag{4.20}$$

Thus, we formulate our optimization problem as follows,

$$\begin{aligned}
&\min && C(y) && \text{(4.21a)}\\
&\text{subject to} && \overline{y} \in \mathcal{Y}. && \text{(4.21b)}\\
& && \mathcal{T}_{ch}(C_m + C_p) \leq \mathcal{B}, \forall m \in \mathcal{M}, && \text{(4.21c)}\\
& && T_{trip} \leq \mathcal{T}_{trip}^{max}, \forall m \in \mathcal{M}, && \text{(4.21d)}\\
& && T_{tot} \leq \mathcal{T}_{char}^{max}, \forall m \in \mathcal{M}, && \text{(4.21e)}\\
& && T_{wait} \leq \mathcal{T}_{wait}^{max}, \forall m \in \mathcal{M}, && \text{(4.21f)}\\
& && \sum_{m=1}^{\mathcal{M}} y_m = 1, \forall m \in \mathcal{M}, && \text{(4.21g)}\\
& && SoC_{curr} - d_m\beta > 0 && \text{(4.21h)}
\end{aligned}$$

where $\mathcal{B}$ represents the maximum budget of EV, while $\mathcal{T}_{trip}^{max}$, $\mathcal{T}_{char}^{max}$ and $\mathcal{T}_{wait}^{max}$ are the maximum travel time, charging time and waiting time respectively. Constraint (4.21c) states that the total charging cost should be less than the defined EV charging budget $\mathcal{B}$. Constraints (4.21d), (4.21e) and (4.21f) ensures that the cost of traveling, charging and waiting time should be within defined thresholds i.e., $\mathcal{T}_{trip}^{max}$, $\mathcal{T}_{char}^{max}$ and $\mathcal{T}_{wait}^{max}$ respectively. Constraint (4.21g) ensures that only one CS should be selected. Finally, constraint (4.21h) states that the EV's remaining SoC should be enough to reach the selected CS.

### 4.5.2 Discussion

It should be noted that the optimization problem will be solved off-chain by the individual EVs as the CS selection decision needs to be made in real time. Solving this optimization problem in smart contract on public blockchain will incur a significant delay as well as storage and transaction cost. After solving the optimization problem off-chain, the EV will make a remote reservation with selected CS through smart contract on public blockchain network as an anonymous entity to ensure the availability of the selected time slots and the real identity of the EV will only be revealed after reaching the selected CS. Thus, EV will select the best-suited CS off-chain, while the protocol to reserve a charging slot, authenticate EV and payment will run on the blockchain network.

As the CS decision is a binary variable and there are integer variables in equality constraints, the optimization problem formulated in Equation. (4.21a) is an Integer Linear Program (ILP). As the optimization problem formulated in Equation. (4.21a) is not bounded by any lower bound therefore, it is trivially solvable. In order to solve this optimization problem, we propose the direct invocation of an available optimization solver such CPLEX.

**Practicality:** Our solution can be easily implemented with a simple client-side software application, which retrieves the charging parameters from different RSUs through an Application Programming Interface (API) in real-time. This information can be used by software application to select the best CS by solving an optimization problem. Moreover, a blockchain client functionality can be easily added in this software application, where a user can send transaction with a valid blockchain address. Similarly, CSs can also be equipped with a software application, which enables them to communicate with EVs with valid blockchain address through smart contract.

## 4.6 Security analysis

In this section, we analyze the privacy and security properties of our proposed protocol in terms of privacy and security requirements explained in Section. 5.2.3. After that, we evaluate the performance of our proposed blockchain-based protocol.

In order to check the security vulnerabilities of our proposed approach we employ STRIDE[4] mnemonic: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. As EV and CS are making reservation and charging payment on blockchain network, no malicious entity can spoof their identity during the charging process. Moreover, as the information is stored in the blockchain network, no one can tamper with the reservation and payment information. EV and CS are signing the transactions with their private keys therefore, they cannot repudiate (claim not to make reservation or charging). As EV is not communicating with any other entity during the CS selection process, no private information of EVs has been disclosed to other entities in the network. Finally, as EV needs to send some initial deposit and CS needs to spend some gas in running the *reservation* function, no entity can spam the reservation request in the network.

Table 4.1 Security comparison

| Functionality | BlockEV | OCPP | ISO15118 | [6] | [36] | [37] | [38] |
|---|---|---|---|---|---|---|---|
| No trusted intermediary | ✓ | *x* | *x* | ✓ | ✓ | ✓ | *x* |
| EV privacy | ✓ | *x* | *x* | ✓ | ✓ | ✓ | *x* |
| Availability guarantee | ✓ | *x* | *x* | *x* | *x* | *x* | *x* |
| Traceability & Accountability | ✓ | *x* | *x* | *x* | *x* | ✓ | *x* |
| Protection against impersonation attack | ✓ | *x* | *x* | ✓ | ✓ | ✓ | ✓ |
| Protection against forgery attack | ✓ | *x* | *x* | ✓ | ✓ | ✓ | ✓ |
| Protection against MITM attack | ✓ | *x* | *x* | ✓ | ✓ | *x* | *x* |
| Protection against tamper attack | ✓ | *x* | *x* | ✓ | ✓ | ✓ | ✓ |
| Protection against DoS attack | ✓ | *x* | *x* | *x* | ✓ | ✓ | *x* |
| Fake reservation attack | ✓ | *x* | *x* | *x* | *x* | *x* | *x* |

[4] https://www.microsoft.com/security/blog/2007/09/11/stride-chart/

### 4.6.1 Security properties

**Privacy of EV:** The foremost security consideration is the EV user's privacy while selecting the CS, i.e, no entity in the network except the selected CS gets private information about EV. Here, the private information of an EV can be specifically described as its exact location, SoC, energy demand, charging routine etc.

**Theorem 1.** BlockEV ensures the privacy and pseudo-anonymity of the participating EVs.

*Proof:* In our proposed protocol, EV is not communicating any information with RSUs or CSs before selecting a CS therefore, no entity in the network can learn about the identity of EV, its current location or energy demand. Specifically, the EV uses its public key $pk_{EV}$ as the pseudonym to communicate with the CS in order to guarantee the anonymity. The messages in the protocol,

$$RSUs \rightarrow EVs : request_i = \{ID_{CS_m}||loc_{CS_m}||T_s||T_e||\alpha||T_w||C_{m_CS_m}||C_{p_CS_m}||Rep_{CS_m}\}.$$

$$EV \rightarrow SC_{reservation} : input = \{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_{sel}||deposit_{init}\}.$$

$$EV \rightarrow SC_{authentication} : input = \{dec_{sk_{EV}}((randomnumber)_{pk_{EV}})\}.$$

$$EV \rightarrow SC_{payment} \rightarrow CS: input = \{T_{start}, T_{end}\}.$$

are transmitted in the network with the associated $pk_{EV}$, thus ensuring anonymity. Moreover, the EV is making a remote reservation with the selected CS on blockchain network therefore, at reservation time the identity of EV is still not revealed to CS. The identity of EV will only be revealed to a selected CS after reaching the CS premises. Finally, when the EV leaves the CS after charging, CS cannot track the position of EV in real time, to predict the charging routine or driving patterns as EV leaves the charging station.

**Mutual Authentication:** After reaching CS premises, EV needs to authenticate itself to CS by showing the secret random number shared between them at the time of reservation. Smart

contract will take random number from EV and CS and after comparing the random number in the $authentication$ function, smart contract will mutually authenticate both entities for the corresponding reservation.

**Theorem 2.** The authentication in the proposed scheme is secure.

*Proof:* : In the proposed protocol, authentication for EV and CS is done with the help of smart contract. After reaching the CS premises, the following messages are exchanged between EVs, CSs and smart contract.

$$EV \rightarrow SC_{authentication} : input = \left\{dec_{sk_{EV}}((randomnumber)_{pk_{EV}})\right\}.$$

$$CS \rightarrow SC_{authentication} : input = \{randomnumber\}.$$

As only EV can decrypt the random number with its private key, no entity in the network can forge the authentication. Finally, as the smart contract function compares the random number $compare(\left\{dec_{sk_{EV}}((randomnumber)_{pk_{EV}})||(randomnumber)_{pk_{EV}}\right\})$ provided by EVs and CSs automatically, no adversary can change this functionality on the blockchain network, thus leading to secure authentication.

**No trusted intermediary:** In our proposed protocol, EVs are communicating and scheduling the charging slot with CS directly through blockchain network and smart contract, unlike the traditional intermediary controller to schedule EVs. All the CSs are working independently and have equal rights to trade energy with EVs based on its available services without any involvement of third party.

**Data integrity and immutability:** Once the reservation has been made between EV and CS and is written on the blockchain network, no entity can modify or tampers with this information. Similarly, the payment and reputation information written on the blockchain network will remain immutable in future and will be helpful in auditability and traceability of the charging and payment information.

### 4.6.2 Security attacks analysis and comparison

In this subsection, we analyze our model against different security attacks. Malicious EV and CS can launch external and internal attack. We now present how different security attacks can be resisted by our proposed protocol.

**Impersonation Attack:** In the proposed protocol, if a malicious EV user tries to impersonate as a legitimate EV user, it needs to make a reservation with a CS through a real blockchain address by sending $\{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_{sel}||deposit_{init}\}$ to the smart contract by signing the transaction by its private key as $\{T_A||SoC_{curr}||SoC_{des}||C_m||C_p||T_{sel}||deposit_{init}\}_{sk_{EV}}$. The knowledge of private key $sk_{EV}$ only belongs to a real EV user, which makes the blockchain addresses and identities unique and cannot be used by any other party except the owner with $sk_{EV}$ thus, a malicious EV cannot impersonate a benign EV.

**Forgery/False data injection Attack:** In the proposed protocol, as the charging rate vary based on the electricity load, a charging station as well as an EV can use fake values at the time of broadcast information and actual reservation. The proposed scheme is able to detect any forgery attempt with the help of smart contract. Lets take price as an example, the smart contract will simply take $\{C_m||C_p\}_{EV}$ and $\{C_m||C_p\}_{CS}$ as an input from both EV and CS. If $\{C_m||C_p\}_{EV} = \{C_m||C_p\}_{CS}$, the reservation function will be executed. Similarly, if $\{C_m||C_p\}_{EV} \neq \{C_m||C_p\}_{EV}$ , the reservation will be terminated by the smart contract. Other values forgery analysis is similar to the above analysis. Thus, the proposed protocol provides protection against CS trying to make reservation with fake/forged information.

**Tamper Attack:** CS can try to change the reservation information once the reservation has been made. One can consider a scenario where the electricity price was different at the time of reservation however, the price changed at the actual charging time slot, thus a monetary incentive can motivate CS to change price information. Fortunately, due to the tamper-proof feature of blockchain, once the reservation information is written on the blockchain, it cannot be changed. EV tamper attack analysis is similar to the above analysis. Thus, the proposed protocol provides protection against tamper attacks thus, preserving data integrity.

**Repudiation Attack:** As seen before, the blockchain transactions are signed by the private keys, which makes the addresses and identities unique. Thus, neither EV nor CS can repudiate the reservation or payment information, once these functions have been processed through a smart contract and written in the blockchain network.

**DoS Attack:** In the proposed protocol, an EV can flood CSs with false charging requests however, in our protocol EV needs to send $deposit_{init}$ to make a reservation with smart contract. The smart contract will not be executed until it verifies $deposit_{init}$ i.e., if the requesting EV has enough funds. Similarly, all the available CSs are registered with RSUs with valid certificates, no fake CS can flood the network with fake broadcast CS information. Thus, the proposed protocol provides protection against the DoS attack.

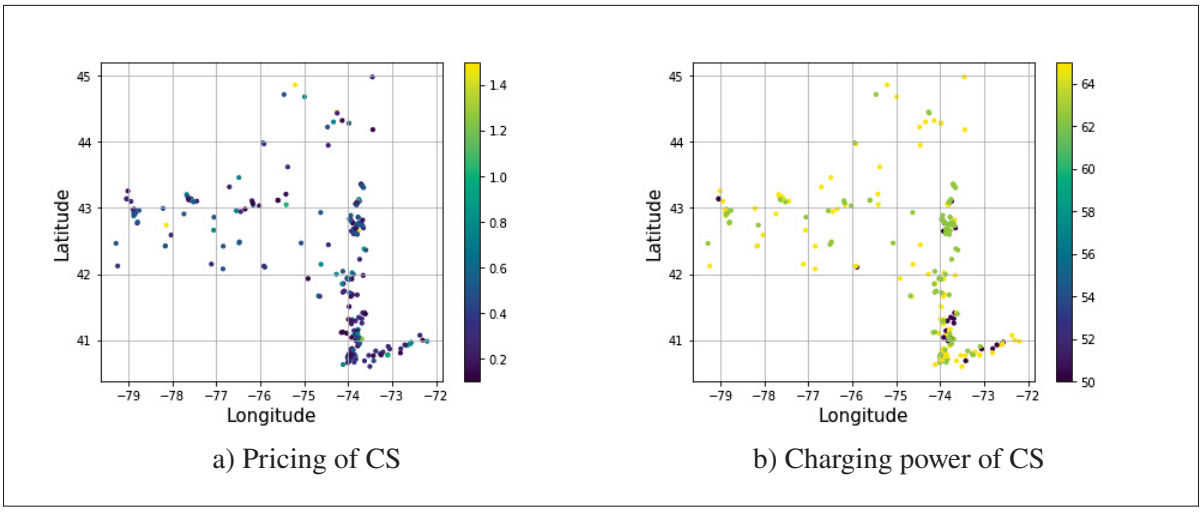
**Man-in-the-Middle (MITM) Attack:** According to the above analysis of tamper-proof data integrity and resistance against repudiation attack, it can be easily assumed that the communication between the EV and CS is verified through smart contract and valid information and messages cannot be forged and modified. Moreover, EV and CS authenticate themselves with the help of freshly generated random number. Thus, our proposed scheme can oppose MITM attack.

**Fake Reservation Attack:** As explained before, EVs need to deposit funds in the smart contract in order to reserve a charging slot at the CS. In case of fake charging slot reservation, where EV wants to reserve all the charging slots of the specific CS, it needs to deposit the funds for each reservation. The deposit functionality makes it hard for a malicious EV to launch a fake reservation attack.

We provide the detailed security analysis comparison of our proposed system with state-of-the-art schemes and industry standard EV charging protocols in Table. 4.1. It can be seen that the proposed scheme offers better security than all the proposed framework. Unlike previous works, one of the novelty of the proposed framework is that it offers the guaranteed availability to the EV, once it has made the remote reservation with the CS. Similarly, with the deposit functionality, the proposed framework mitigates the fake reservation at the CS.

## 4.7 Numerical results

In this section, we evaluate BlockEV in terms of performance and price. Firstly, we analyze the computation time to solve the optimization problem and end-to-end delay, i.e., the time to run the complete protocol. We then perform the price analysis for the proposed protocol and compare it with them nearest CS strategy. Moreover, we also analyze the blockchain storage and transaction overhead of the proposed design and compare it with state-of-the-art solutions. Finally, we provide the gas cost estimation for the smart contract functions, i.e., the price associated with calling a smart contract functions.




a) Pricing of CS b) Charging power of CS

Figure 4.4 Blockchain analysis

### 4.7.1 System setup

We run our experiments on $8^{th}$ Generation Intel $Core^{TM}$ i7-8650U Processor with 8 GB RAM. We have implemented our prototype design in Python programming language using CPLEX Python API to validate the feasibility and effectiveness of the proposed protocol. We implement blockchain functionality through Go language implementation (Geth) and web3py (*https://web3py.readthedocs.io/en/stable/*), a Python library to interact with Ethereum network. Moreover, we use the Solidity programming language to implement the smart contract.

**Dataset and simulation settings:** In the work, we have used a real dataset from US department of energy website. We have considered the fast EV charging stations in New York as shown in Fig. 4.5a. The dataset includes the CS location, ID, pricing information etc. We also added the charging power of the CSs in our dataset specific to the charging network. The waiting time at the CSs is chosen randomly and it varies from 10 sec to 20 sec (Bryden, Hilton, Dimitrov, de León & Cruden, 2019) as this information was not available. In our experiments, we considered the number of CSs ranging from 2 to 50, based on different experimental scenarios and settings. The default value of transforming the waiting time, travelling time and charging time to monetary cost is set to 0.1$ per minute. In addition, to perform the experiments, we randomly selected the position of EV from the map shown in Fig. 4.5a. We calculate the travelling time to a certain CS by assuming the average velocity of EV to be 50km/h. We repeat our experiments 1000 times and represent our results in average of these experiments. Finally, we plot the pricing and charging power dataset against the location of CSs in order to get the distribution insights. It can be seen from Fig. 4.4a and Fig. 4.4b that the pricing and charging power is scattered over all the available CSs. In our experiments, we select different random locations of EVs and the results are averaged out.

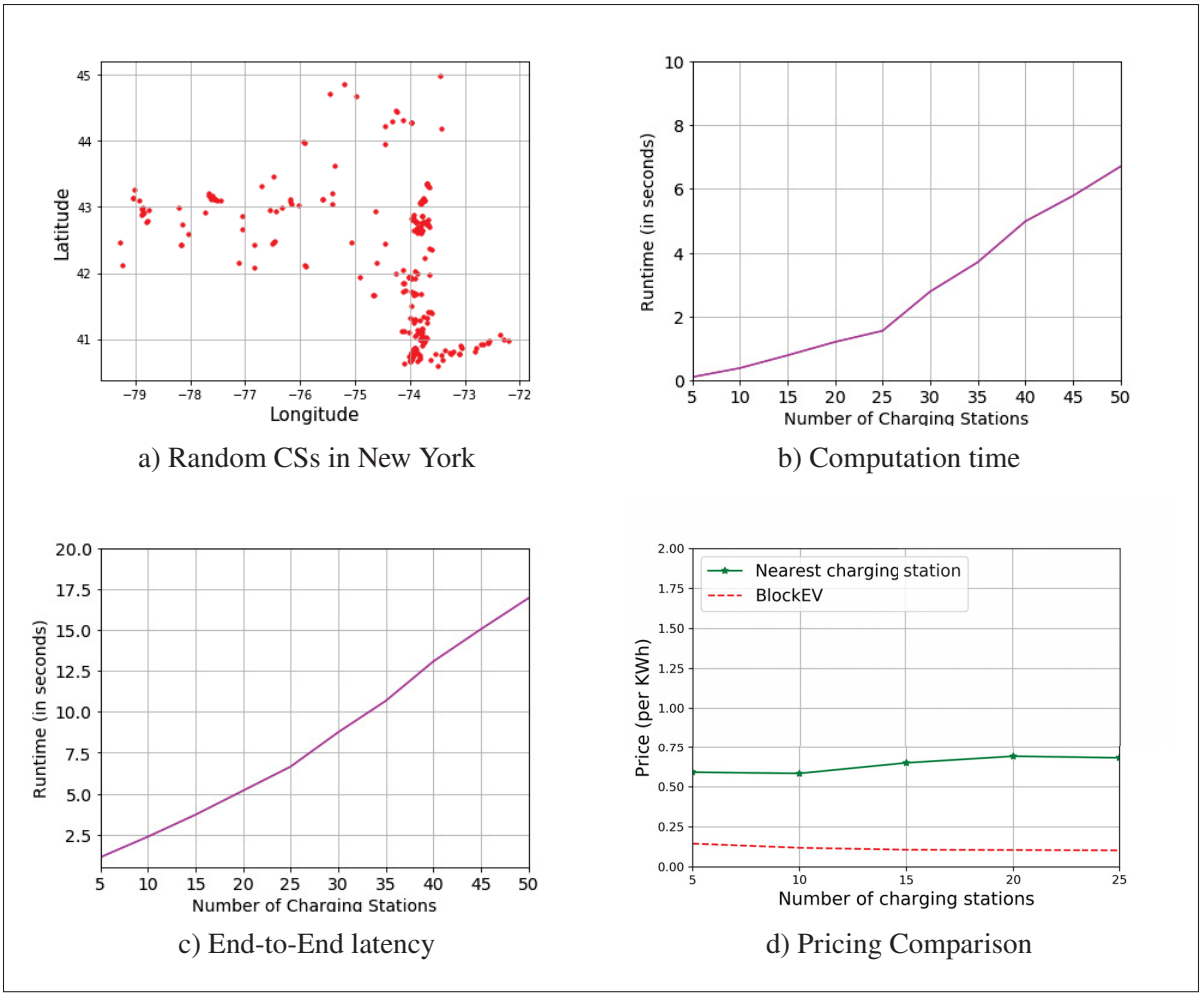
### 4.7.2 Results

**Computation time:** We first investigate the CS selection mechanism in terms of computation runtime in Fig. 4.5b. We implement the CS selection optimization problem formulated in Equation. (4.21a) in the CPLEX using Python CPLEX API. We run these experiments several time for different number of CSs to plot the average runtime values. It can be seen in Fig. 4.5b that the computation time is low when EV needs to select a CS with small search space, i.e., small number of CSs. However, increasing the number of CSs results in more time to generate a solution compared to small number of CSs. One can further reduce the computation run times with the help of powerful computing servers as well as through optimized programming implementation.

We then perform the end-to-end latency analysis, i.e., the time when the CSs send the information to RSUs till the EVs make the CS selection decision. This analysis is shown in Fig. 4.5c. We perform this experiment by creating an API using Python-flask framework. This API generates the data instances for each CS based on the real dataset explained above. This data includes the CS ID, location, price, waiting time, charging power and available time-slots. It can be seen in Fig. 4.5c that as the number of CSs increase, the overall end-to-end latency increases. It should be noted that this end-to-end latency analysis does not include the time to mine the block as this depends on the type of blockchain used.

**Pricing analysis:** We now investigate the impact of proposed CS selection mechanism on the overall price and compare it with the price offered by nearest CS. We perform this experiment 1000 times and represent our results in average of these experiments. Fig. 4.5d shows the comparison of the price per kWh for different number of CSs. It can be seen in the Fig. 4.5d that BlockEV offers less price compared to the price offered by the nearest CS. This is because the solution to the optimization problem defined in Eqn. 4.21a will ensure that the CS with minimum price is selected among the solution space. Thus, BlockEV solution offers least price in all cases.

**Blockchain Transaction Overhead:** We further investigate the transaction overhead of the proposed protocol in Fig. 4.6a and Fig. 4.6b, in order to compare the total number of transactions with the baseline (work in (Knirsch *et al.*, 2018)). Our proposed protocol incurs 4 transactions for one run of the protocol, therefore for $\mathcal{G}$ number of EVs, the total number of transactions will be $4\mathcal{G}$. In (Knirsch *et al.*, 2018), the total transaction overhead for one run of protocol is $2M + NMn$, where M, N and n are number of EVs, CSs, and bids respectively. Fig. 4.6a shows the comparison of transaction overhead for different number of EVs with $M = 5$ and $n = 1$. It can be seen that our proposed protocol incurs less transaction overhead compared to the work in (Knirsch *et al.*, 2018). This is because the protocol in (Knirsch *et al.*, 2018) proposes each CS to send a bid transaction thus, increasing transaction overhead. Similarly, Fig. 4.6b shows the comparison with respect to number of CSs. It can be seen that increasing CSs does not affect




a) Random CSs in New York

b) Computation time

c) End-to-End latency

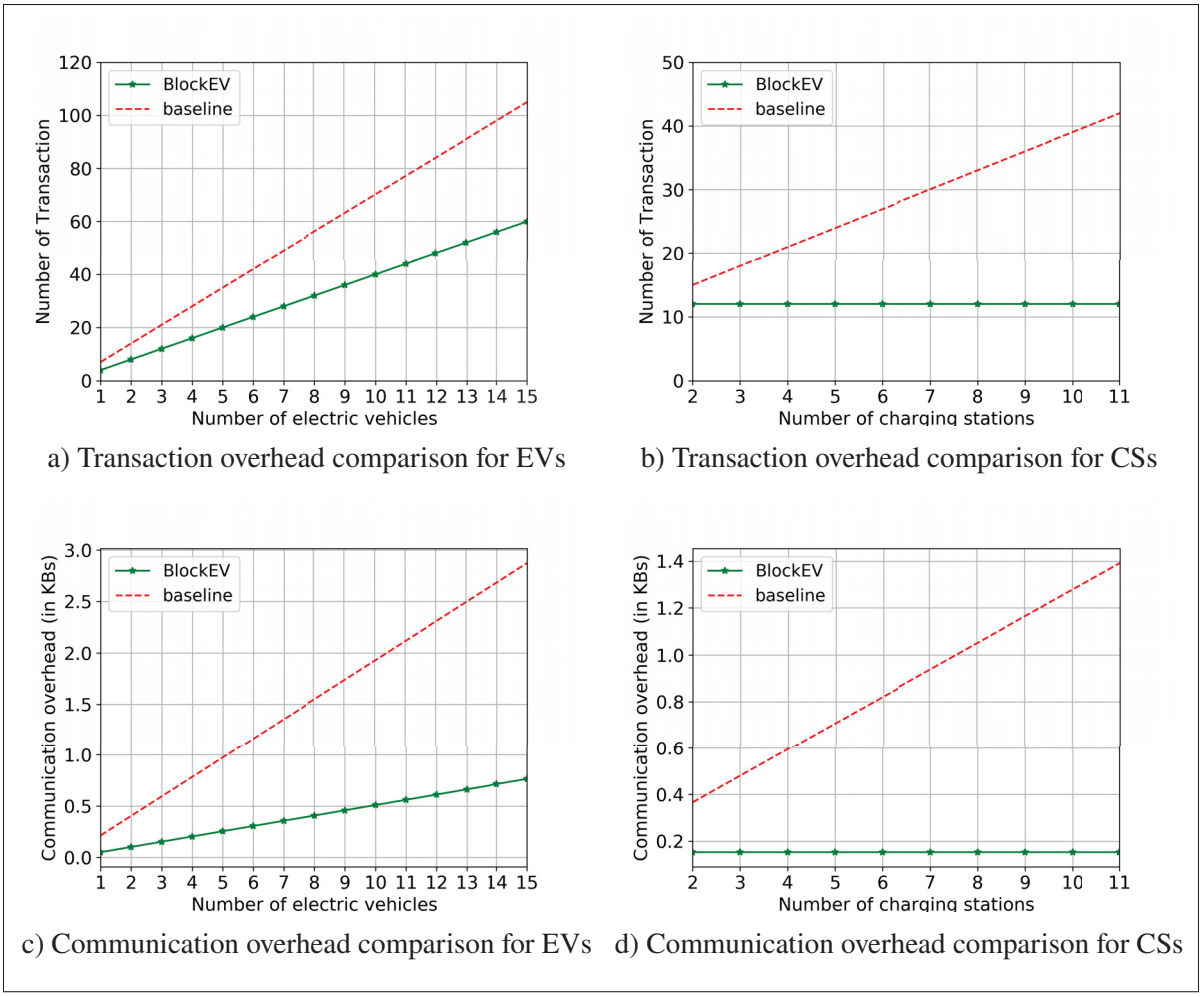
d) Pricing Comparison

Figure 4.5 Performance and pricing analysis

the transaction overhead because in our protocol, only single CS is taking part in blockchain transaction unlike (Knirsch *et al.*, 2018), thus providing better scalability.

**Blockchain storage overhead:** In this experiment, we compare the overall storage overhead in the blockchain network with the baseline (work in (Knirsch *et al.*, 2018)) in Fig. 4.6c and Fig. 4.6d. In the reservation function, $T_A$ and $T_{sel}$ can be represented by 2 bytes[5] each. $C_m$ and $C_p$ can be represented by 2 bytes each (Knirsch *et al.*, 2018), while $SoC_{des}$ and $SoC_{curr}$ can be represented by 1 byte each. Random number can be represented by 2 bytes. Similarly, the

[5] https://www.ibm.com/support/knowledgecenter/




a) Transaction overhead comparison for EVs b) Transaction overhead comparison for CSs

c) Communication overhead comparison for EVs d) Communication overhead comparison for CSs

Figure 4.6 Blockchain analysis

authentication transaction can be represented by 2 bytes, while $T_{start}$ and $T_{end}$ in the payment transaction can be represented as 2 bytes each. Finally, the reputation transaction can be represnted by a 30 bytes ID (Knirsch *et al.*, 2018) and 1 byte for rating. In (Knirsch *et al.*, 2018), the total storage overhead for one run of protocol is $8 + 32Nn + 16$ bytes, where N and n are number of CSs, and bids respectively. Fig. 4.6c shows the storage overhead comparison for different number of EVs. It can be seen that BlockEV incurs less storage overhead compared to the baseline. Similarly, Fig. 4.6d shows the storage overhead comparison with respect to the number of CSs. It can be seen that increasing CSs does not affect the storage overhead because in BlockEV, EV is communicating with only single CS at a time unlike (Knirsch *et al.*, 2018).

We now provide a detailed transaction and storage overhead comparison of our proposed solution with other schemes. We assume an EV charging network with N = 1000 EVs and M = 5 CS to compare the transactions incurred by different schemes over a period of 1 day as shown in the Table. 4.2.

Table 4.2 Blockchain transaction and storage comparison

| Schemes | Transaction overhead | Transaction per day | Storage per day |
|---|---|---|---|
| BlockEV | $4M$ | 4000 | 784KB |
| (Knirsch *et al.*, 2018) | $2M + NMn$ | 7000 | 1.288MB |
| (Gao *et al.*, 2018) | $10M$ | 10000 | 5.36MB |

**Storage overhead for EVs and CSs:** In this work, we have considered Ethereum blockchain to allow EVs and CSs to communicate through smart contracts. Thus, the private and public keys (32 bytes and 64 bytes respectively) along with blockchain address (20 bytes) need to be stored by both EVs and CSs in order to take part in EV charging protocol. Moreover, in order to solve the decision optimization problem, EV needs to temporarily store the data to find the best-suited CS. The RSU will broadcast the CS charging information to the EV which contains the CS ID (30 bytes), location (6 bytes), start time of available charging slot (2 bytes), end time of available charging slot (2 bytes), charging power (2 bytes), estimated waiting time (2 bytes), charging cost per unit time (2 bytes), parking cost per unit time (2 bytes) and reputation (1 byte). Adding them will result in a total of 49 bytes of broadcast information for one CS. Thus, the data storage requirement for problem depends on the number of CSs. To generalize, the temporary data storage requirement to solve the decision optimization problem can be represented as $49\mathcal{M}$, where $\mathcal{M}$ is the number of CSs.

**Smart contract analysis:** In order to evaluate the smart contract, we provide the gas cost estimation of the functions. Gas cost is a unit that measures the amount of computational effort that smart contract will take to execute certain operations. In order to calculate the gas cost, we invoke the smart contract function with given input values. Once the transaction is added to the blockchain network, we find the transaction with the help of transaction hash and find the gas cost consumed to run the said function. As explained before, the smart contract includes

four main functions: reservation, authentication, payment and reputation. Both EV and CS send related information to the *reservation* and *authentication* function and the function is executed and a transaction is written in the blockchain network if the information is matched for both EV and CS. Once the charging is done, *payment* function takes total charging time information from the smart meter and transfers the coins from EV to CS. Finally, EV rates CS through the *reputation* function based on its user experience. We provide gas cost estimation for functions in Table. 4.3.

Considering the gas cost and the associated dollar fee, one can say that this blockchain fee is too expensive for EV charging. However this problem can be solved by using the blockchain with low transaction fee, since our design can work with any blockchain with smart contract functionality.

Table 4.3 Gas cost estimation for smart contract functions

| **Function** | **Gas Units** | **Gas Price (in Wei)** | **Gas Cost (in Ethers)** |
|---|---|---|---|
| reservation | 35030 | 1000000000 | 0.00003503 |
| authentication | 24750 | 1000000000 | 0.00002475 |
| payment | 47900 | 1000000000 | 0.00004790 |
| reputation | 31390 | 1000000000 | 0.00003139 |

## 4.8 Conclusion

The untrusted centralized nature of energy markets and EV charging infrastructures results in several privacy and security threats to EV user's private information. In this work, we propose BlockEV, a blockchain-based efficient CS selection protocol for EVs to ensure the security and privacy of the EV users, availability of the CSs and enhanced overall EV user's experience. First, a blockchain-based framework is introduced to implement secure charging services and trusted reservation for EVs with the execution of smart contract. Second, we theoretically formulate the CS selection decision optimization problem. Evaluations show that our proposed system incurs less blockchain transaction and storage overhead compared to existing schemes

along with improved scalability. Moreover, we plan to extend this work by introducing the privacy-preserving aspect in order to provide full privacy to the EV users and the CSs.

# CHAPTER 5

# A BLOCKCHAIN-BASED PRIVACY-PRESERVING CHARGING STATION RESERVATION AND PAYMENT SCHEME FOR ELECTRIC VEHICLES

Syed Muhammad Danish[1] , Munim Shabir[1] , Kaiwen Zhang[1] , Hans-Arno Jacobsen[2] , Syed Ali Hassan[3]

[1] Département de génie logiciel et des TI, École de Technologie Supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Electrical & Computer Engineering, University of Toronto
[3] National University of Sciences and Technology (NUST), Pakistan

Paper submitted to *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, June 2022

This chapter describes our total order extension to content-based publish/subscribe systems. Section 5.1 provides an introduction to security and privacy problems in EV charging network. Section 5.2 introduces a blockchain-based EV charging architecture. Section 5.3 presents a ring signature-based privacy-preserving reservation and payment scheme for the EV charging network. Section 5.4 presents the CS information verification mechanism. Section 5.5 presents the dispute resolution mechanism among EVs and CSs. Section 5.6 presents the security analysis of the proposed protocol. Section 5.7 presents the evaluation and numerical results for the proposed protocol.

## 5.1 Introduction

Electric vehicles (EVs) are among the emerging intelligent transportation system (ITS) applications in smart cities and have been recognized as a promising solution to reduce $CO_2$ emissions compared to traditional gasoline vehicles (Goli & Eskandarian, 2014; Zhou *et al.*, 2017; Blum & Eskandarian, 2007; Pajic *et al.*, 2018). With the number of EVs increasing rapidly, by 2030 it is estimated that there will be 250 million EVs available annually (Liu *et al.*, 2019). Because of this dramatic growth, EV management may face many new challenges and therefore, to provide vital comfort for EV drivers, there is a need to manage EV charging efficiently.

Current charging networks allow EVs to communicate with charging stations (CSs) through industry-standard protocols, such as the open charge point protocol (OCPP) and ISO 15118 (Antoun *et al.*, 2020). These protocols are, however, not secure and pose serious privacy and security risks. A trusted mobility operator facilitates the ISO 15118 protocol by performing authentication and identification of EVs by maintaining their private information (Eiza *et al.*, 2018; Bao *et al.*, 2018). Private information includes EV location, availability, identity, charging and payment information, etc. Moreover, it also performs EV tracking to direct them to a desirable CS, which raises serious concerns as EVs' private information can be leaked unintentionally or intentionally (Bao *et al.*, 2018). Trusted central management reserves a charging slot for EVs through the OCPP charging protocol by sharing EVs' private information with selected CSs (Antoun *et al.*, 2020). In addition, despite the transport layer security (TLS) assurances in the OCPP protocol, it is vulnerable to impersonation attacks where an attacker can pose as a CS to ask for the EV's private data (Antoun *et al.*, 2020).

The tracking of an EV and collection of the private data associated with it by a trusted centralized intermediary raises serious privacy and security concerns, as these private data can be leaked/sold or used for malicious activities (Knirsch *et al.*, 2018; Au *et al.*, 2013). EV locations may be exploited to target advertising by malicious businesses (Knirsch *et al.*, 2018; Cao *et al.*, 2015). Similarly, attackers can use the EV owner's private information to predict behaviors such as workplace, working hours, and other important patterns as well as to track positions (Knirsch *et al.*, 2018; Clarke & Wigan, 2011). Moreover, these trusted intermediaries are susceptible to single points of failure. These centralized EV charging protocols pose serious security and privacy concerns for EVs as they require EVs to divulge private information to central management and central scheduling to reserve a charging time slot. Because of the aforementioned security and privacy concerns, a decentralized EV charging reservation protocol is needed, in which EVs can communicate and reserve charging slots at the CS without revealing their personal information while maintaining their privacy.

We propose the use of blockchain-based distributed ledger technology (DLT) (Pajic *et al.*, 2018; Li *et al.*, 2019), to address the above-mentioned privacy and security problems. In the existing

literature, privacy and security aspects of EV charging reservations are largely overlooked (Knirsch *et al.*, 2018; Danish *et al.*, 2020c). As soon as the EV arrives at CS premises for charging, its physical identity is exposed, i.e., its blockchain address can be linked with its physical identity. The privacy of EV owners is particularly at risk due to the possibility of personal information being leaked or sold to predict their daily habits and track their location. Additionally, EV charging history and payment information can be retrieved using the exposed blockchain address in a public blockchain setting. Therefore, it is necessary to ensure end-to-end privacy preservation for EV owners. Additionally, the charging service payments via smart contracts may disclose the blockchain address of the payer (EV) so that the CS can correlate it with the physical identity of the payer. Therefore, it is also necessary to provide a design, that hides the identity of the payer (EV) for smart contract-based payment mechanisms.

Furthermore, the existing works do not contain a mechanism for verifying whether charging slots at CSs are available. To reserve a charging slot, EVs must trust the availability information provided by CSs, which could lead to a reservation conflict. Moreover, there are no mechanisms for resolving conflicts between EVs and CSs, and current works do not prevent disputes, such as if a CS does not comply with the agreed service requirements or an EV does not pay the required amount to a CS. As a result, there is a need to verify CSs information and resolve conflicts between EVs and CSs without involving trusted third parties

In this paper, we propose a blockchain-based private CS reservation protocol that allows EV owners to reserve a charging slot privately, without sharing their personal information or exposing their blockchain addresses at the CS. EVs collaborate using ring signatures (Rivest *et al.*, 2001) to anonymously make a reservation at selected CS. To enforce charging reservations between EVs and CSs, blockchain smart contracts are used. Additionally, a secure smart-contract-based payment mechanism is implemented using a time-lock protocol, which aggregates the payer's transactions into an obfuscated public ledger, such that no outside observer can identify the exact source of a blockchain transaction. The benefits provided by a blockchain include anonymity, immutability, availability, transparency, and the ability to maintain trust in an untrusted environment without a trusted third-party intermediary.

In addition, the CS information verification protocol is implemented with the help of homomorphic encryption based secure multiparty computation (SMC), where EVs collaboratively verify the available and occupied charging slots at nearby CSs by sharing their reservation information with each other, given that no EV learns about the actual reserved charging slot information of the other EVs. Finally, we propose a mechanism based on trusted execution environments (TEEs) (Al-Bassam, Sonnino, Król & Psaras, 2018) that directly feed data into smart contracts to resolve disputes. Data integrity is preserved through the execution and attestation of TEEs, which are tamper-proof.

The proposed protocol aims to ensure EV anonymity and physical identity unlinkability on the blockchain network in an untrusted environment along with the secure payment and conflict resolution mechanism without the need for any trusted third party. This paper has the following contributions:

1. We propose a blockchain-based end-to-end privacy-preserving CS reservation protocol, which enables EVs to reserve a charging slot at the selected CS privately, without sharing their private information with other entities in the network. Unlike previous protocols, the proposed protocol protects the EV's privacy by dissociating its real identity from the blockchain address, thereby preserving the EV's privacy.
2. We propose an SMC-based CS information verification protocol that allows EVs to collaboratively verify the availability of charging slots from the CS by securely sharing reservations in an untrusted environment. To the best of our knowledge, this problem has not been addressed in previous works.
3. Based on time-lock deposit protocols, we propose a fair payment smart contract design between EVs and CSs. EV drivers' payments are fed into the smart contract, which aggregates the balance in such a way that no CS is able to link charging service payments with blockchain addresses, while still ensuring that CSs receive full payment for the services they provide.
4. We propose a dispute settlement mechanism utilizing TEEs, where charging information from EVs and CSs is securely fed to the dispute resolution smart contract. In the case of a

possible conflict, the mechanism imposes a penalty for abnormal behavior both by EVs and CSs.

5. We provide a detailed security analysis of the proposed protocol along with a detailed comparison with industry standards and state-of-the-art schemes. To show the effectiveness of the proposed scheme, we analyze the proposed protocol under different security threats.
6. We provide numerical results for the proposed protocol to demonstrate its effectiveness by comparing it with other conventional schemes.

## 5.2 EV charging reservation problem

This section proposes a blockchain-based privacy-preserving EV charging network comprised of CSs, EVs, RSUs, and a blockchain network. The EV charging network architecture is shown in Fig. 5.1.

### 5.2.1 System model

We consider a network of $\mathcal{M}$ CSs from different competitive operators along with $\mathcal{R}$ RSUs, deployed geographically in a city. $\mathcal{M}$ is a set of CSs, where $m \in \mathcal{M}$ represents an arbitrary CS. Similarly, $\mathcal{R}$ is a set of RSUs, where $r \in \mathcal{R}$ represents an arbitrary RSU. A network of $\mathcal{G}$ battery-operated EVs is also operating in the city, where $g \in \mathcal{G}$ represents an arbitrary EV. Every CS broadcasts its charging information to the $\mathcal{R}$ RSUs and every $g \in \mathcal{G}$ receives this broadcast information of CSs through the $r^{th}$ RSU in their vicinity. Each $g$ needs to recharge its battery for daily operations and using this broadcast information, it independently selects a CS to recharge its battery (Danish *et al.*, 2020c). Once the EV selects a CS based on its service requirements, it makes a remote reservation at the selected CS for the required charging time slot. Once the reservation is made, the EV goes to the CS premises to charge itself. EVs, RSUs, and CSs are public entities, connected to a public blockchain network with smart contract functionalities.

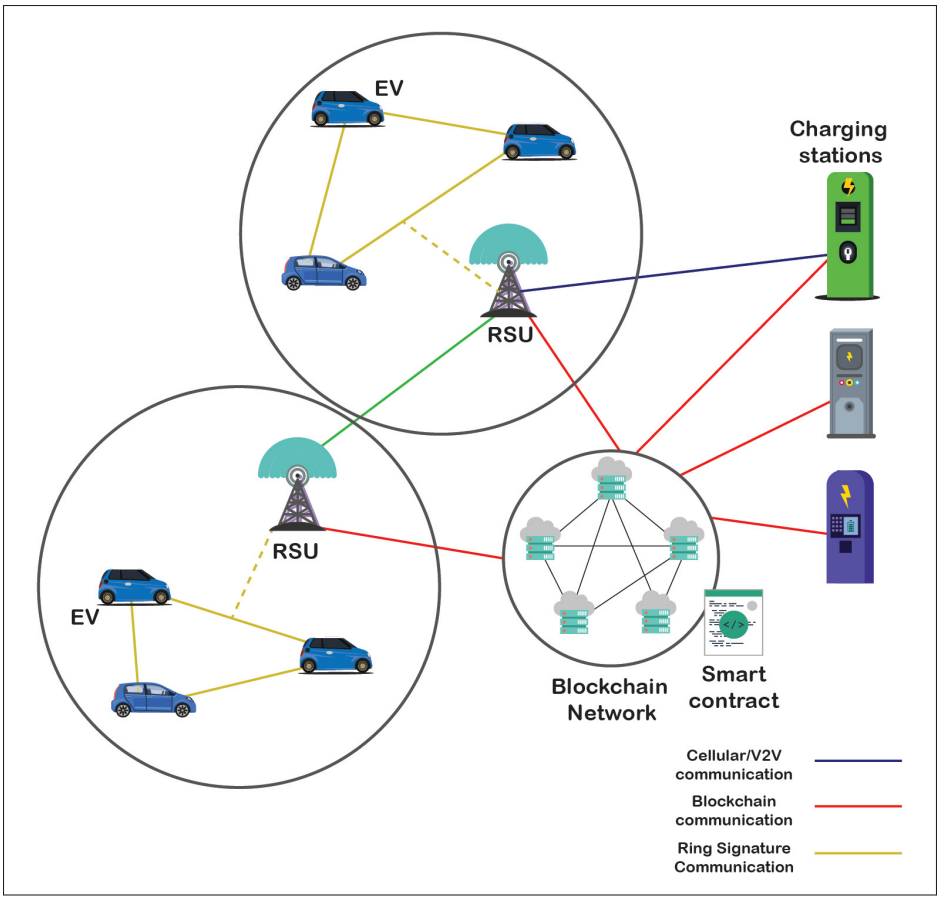



Figure 5.1 Blockchain-based privacy-preserving EV charging network

### 5.2.2 Problem formulation

The main requirement when operating in a public environment is to ensure the privacy and anonymity of the EV's owner during the protocol's communication. Prior works provided partial anonymity and privacy, such that EV owners' identities were revealed because of the links to blockchain addresses, which exposed the history of EV charging and payments. In light of the above security issues, we wish to ensure EV privacy in two ways. First, no entity within or outside the network is allowed to access an EV's private information or link any information to an EV. In addition, once an EV has been charged and leaves the CS premises, no entity in or outside the network can associate the EV's physical identity with the blockchain address. In summary, the research goals can be formally defined as follows.

**Goals:**

- RSUs, CSs, or any external entity cannot link the reservation information to a certain EV, i.e., the reservation mechanism should be anonymous and unlinkable.
- At CS premises, CS should not be able to link the blockchain address of EVs with the physical identity of EV owners.

To achieve the goals presented above, we propose the use of ring signatures to ensure EV's anonymity and end-to-end privacy-preservation of EV's identity. $\mathcal{G}$ EVs in the vicinity of any given RSU collaborate using ring signatures to anonymously make a reservation at the selected CS. RSUs and CSs are directly connected to a public blockchain network with a public account to interact with the smart contract. EVs communicate with RSUs with the help of ring signatures and subsequently, RSU makes reservations at CS on EV's behalf on the blockchain network.

In this work, we consider a VANET communication technology to disseminate the information from CSs to EVs through RSUs with a reliable communication channel and consider a publisher/subscriber (P/S) push model where a nearby RSU (publisher) sends the broadcast information from CSs to the EVs (subscriber), within its radio coverage, without any explicit query (Danish *et al.*, 2020c). Since P/S VANET dissemination is not the focus of our work, we refer the readers to (Cao *et al.*, 2015) for detailed information. Furthermore, we consider a CS information broadcast scenarios, where each CS is connected to RSUs via a reliable communication channel and each CS periodically submits its charging parameters to the RSUs (Danish *et al.*, 2020c). These charging parameters include the cost of charging, available charging slots, estimated waiting time, and charging power. This CS charging information is then broadcast to the nearby EVs by the corresponding RSU.

### 5.2.3 Privacy and security requirements

The main aim of this work is to ensure the end-to-end privacy of the participating EVs in charging slot reservation and charging information verification protocols. The security and privacy requirements need to be fulfilled during the execution of the proposed protocols.

- No participant taking part in the protocol obtains the EV's private information.

- No participant taking part in the protocol obtains information about the EV's blockchain address.
- No participant in the protocol links the EV owner's physical identity to the blockchain address.

### 5.2.4 Threat model

This work considers rational and malicious participants that may threaten EV security and privacy in the EV charging network. We define the roles of different adversaries in EV charging networks as,

- *Rational CS:* We consider CS as a rational adversary in our proposed system, i.e., it can try to gather private information on EVs at the time of EV reservation and charging. CSs can sell EVs' private information to maximize their gain. Moreover, they can broadcast incorrect charging slot information to lure EVs to their premises to learn about their private information.
- *Malicious EV:* A malicious EV may reserve a charging slot at a given CS but never show up at the time of reservation. Moreover, EVs may agree for charging services from selected CSs and later refuse to pay the incurred charging cost to CSs.
- *Honest but Curious RSUs:* Unlike previous works, we assume an honest-but-curious RSU approach, where the RSU operates honestly however, it wants to learn about the private information of EVs and it can leak this sensitive private information related to EVs.

Finally, we assume that the security protocols and cryptographic primitives are inherently secured and cannot be compromised.

## 5.3 Proposed reservation scheme

This section explains the blockchain-based privacy-preserving reservation and payment scheme, which allows EVs to make a reservation and payment at the selected CS anonymously. An EV that wants to make a reservation at CS communicates with the RSU with the help of ring

signatures and subsequently, the RSU helps the EV to make a reservation with the CS on the blockchain network. In this way, the privacy and anonymity properties of EVs are ensured against RSUs and CSs as CS will see the blockchain address of RSUs instead of EVs. A Monero blockchain provides ring signatures-based anonymous transactions however, we do not use Monero blockchain for EV reservation because it does not support smart contracts, and the reservation information needs to be executed in the smart contract environment after checking the validity of information provided by the CS. However, Monero can be used for charging service payment between EVs and CSs. Fig. 5.2 shows the interaction among entities in a privacy-preserving EV charging network. The CS selection mechanism has been covered in (Danish *et al.*, 2020c). Before we provide the detailed specification of the proposed privacy-preserving reservation and payment protocol, we present the preliminary step of CS registration and public/private key generation for EVs and CSs.

### 5.3.1 System initialization

**CS registration:** CSs are public entities, deployed geographically to provide charging services to EVs. CSs broadcast their charging parameters to the RSUs, therefore, all the CSs will first register themselves at the RSUs by generating a set of public ($pk_{CS_m}$) and private keys ($sk_{CS_m}$). After receiving the $ID$, public key, and digital certificate ($Cert_{CS_m}$) information from CS, RSUs will validate it, and eventually, it will be added to the list of available CSs. The registration step is necessary as it will ensure that only authentic CSs register their charging offers at RSUs, thus enhancing security.

Once the CSs are registered, they start sending their charging offers along with the charging parameters to EVs via RSUs. EVs use this charging parameter information to select the best CS among all (Danish *et al.*, 2020c). In our previous work, we have covered the charging station selection problem, and this work focuses on the privacy-preserving charging slot reservations at a selected CS after the CS selection by EVs. The protocol consists of three parts, i.e., reservation, authentication, and payment.

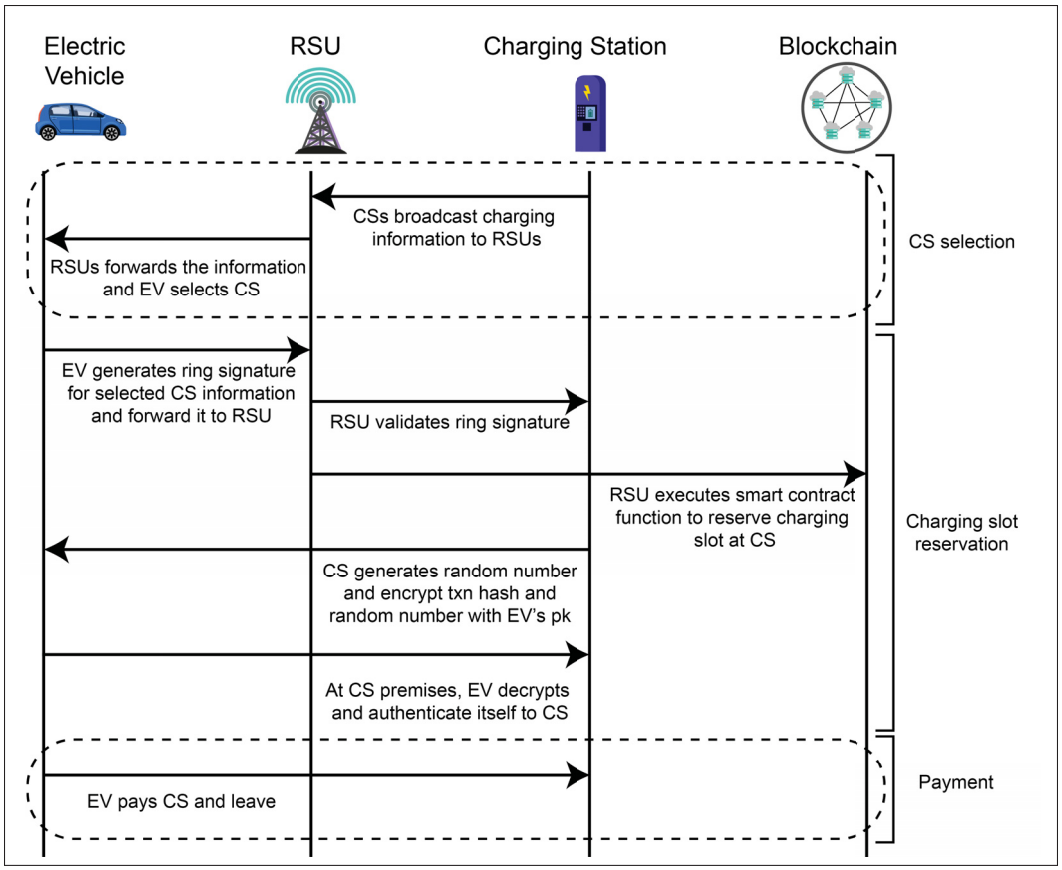



Figure 5.2 Privacy-preserving CS reservation protocol

**Key generation:** First, RSUs will generate their set of private and public keys independently by $\text{Gen}(1^{\lambda}) \rightarrow (pk^{i}_{RSU}, sk^{i}_{RSU})$, by employing elliptic curve cryptography (ECC), where $pk$, $sk$, and $\lambda$ represent the public key, private key and security parameters, respectively. After that, RSUs will share their public keys with the respective EVs along with their IDs and digital signature certificates. This way, EVs can verify the identity of RSUs with which they communicate.

Similar to RSUs, EVs will also generate their set of private and public keys independently by $\text{Gen}(1^{\lambda}) \rightarrow (pk^{i}_{EV}, sk^{i}_{EV})$, using ECC. These public keys will be shared among EVs to use for ring signatures.

### 5.3.2 Reservation

**EV-RSU communication channel:** In this protocol, we employ a ring signature scheme among EVs while communicating with the respective RSU. In our framework, we consider a RSU as an honest but curious entity. Therefore, it is necessary to hide the EV identity. EVs who want to make a reservation at CS collaborate with other EVs in the vicinity of the RSU to hide their identity. RSU can verify the signer EV information easily from the ring signature and use this information to reserve a charging slot at CS.

**Signature Generation:** Let $G = (EV_1, EV_2, ..., EV_n)$ be a list of $n$ ring users' public keys. A vehicle $EV_S$ (S denotes the signer) that wants to communicate with the RSU to reserve a charging slot at CS uses a message $M$, its private key $sk_S$, and $G$ to produce a signature $(M, \sigma)$ as shown in Algorithm. 5.1. $M$ is the selected reservation information, i.e., CS ID, charging price, charging slot time, which is shared with the RSU in order to make a reservation along with the deposit proof, which will be explained in Section V-A. EV selects a CS from a given list of available CSs (Danish *et al.*, 2020c) and uses the selected CS information as message $M$. At the basic level, the signatures will be generated as in Algorithm. 5.1. More details about the generation of ring signature are as follows:

1. $EV_S$ that wants to make a reservation first selects $N$ public keys to form a set $G = (pk_{EV_1}, pk_{EV_2}, ..., pk_{EV_n})$.
2. $EV_S$ then computes a key from $M$ and $\mathcal{G}$ using a publicly known hash function:

$$k = h(M||(pk_1, pk_2, ..., pk_n))$$

3. $EV_S$ then picks a binary random glue value $v$, where,

$$v \in \{0, 1\}^b$$

4. $EV_S$ then chooses $n - 1$ random numbers $x_i \in \{0, 1\}^b$ for all the available public keys in $\mathcal{G}$ and computes $y_i$ by encrypting $x_i$ with the corresponding public key in $\mathcal{G}$. At this point,

signer does not generate its own $x_s$. Set $y_i$ is calculated as,

$$y_i = E_{pk_i}(x_i)$$

, where $y_i = \{y_1, y_2,..., y_{s-1}, y_{s+1},...,y_n\}$ and $x_i = \{x_1, x_2,..., x_{s-1}, x_{s+1},...,x_n\}$. It should be noted that $EV_S$'s $x_s$ and $y_s$ are not included in these sets at this point and will be calculated in the following steps.

5. To find $y_s$, a combining function is solved as follows,

$$C_{k,v}(y_1, ..., y_n) = E_k(y_n \oplus (...E_k(y_s \oplus (...E_k(y_1 \oplus v))) = v$$

6. $EV_S$ then calculates $x_s$ by using its own private key $sk_S$ to decrypt $y_s$ as,

$$x_s = Decrypt_{sk_S}(y_s)$$

7. Once these calculations are performed, the ring signature $\sigma$ can be represented as,

$$\sigma = \{pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n\}$$

Once the signature is calculated by $EV_S$, it will be sent to the corresponding RSU along with $M$ for signature verification and charging slot reservation.

**Signature Verification:** After the ring signature is generated by the $EV_S$, the corresponding RSU will receive $\sigma = \{M; pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n\}$ for verification and after successful validation, it will reserve a charging slot for EV. At the basic level, the signature verification is performed as in the Algorithm. 5.2. More details about the verification of a ring signature are as follows:

1. RSU will compute a key using $M$ and all the available public keys using a publicly known hash function as,

$$k = h(M||(pk_1, pk_2, ..., pk_n))$$

2. With the help of all the available public keys in $\sigma$, the RSU will compute $y_i$ by encrypting $x_i$ with the corresponding public keys as,

$$\{y_1, y_2, ..., y_n\} = E_{pk_i}\{x_1, x_2, ..., x_n\}$$

3. After calculating the values $y_i$, the RSU will verify the correctness of the following equation

$$C_{k,v}(y_1, ..., y_n) = E_k(y_n \oplus (...E_k(y_s \oplus (...E_k(y_1 \oplus v))) = v$$

4. If the equation holds true, the ring signature verification algorithm will return 1 and subsequently, the RSU will make a charging slot reservation at CS using a smart contract.

The ring signature generation is set-up free, i.e., the signer does not need the assistance of the other ring members to put them in the ring; all he needs is knowledge of their regular public keys (Rivest *et al.*, 2001). Therefore, even under the dynamic vehicle movement where the vehicle corresponding to the selected public key is moving far away from the CS, or has entered in the vicinity of another RSU will not affect the ring signature's process. Because, in the end, the signature generator is sending the list of public keys to the verifier along with the signature.

**CS reservation:** Once the ring signature is verified by the RSU, it will call the smart contract function to reserve the corresponding charging slot at the selected CS. Similarly, to hide an EV's identity and dissociate the EV's physical identity from the blockchain address, the RSU is making a reservation at the CS on behalf of the EV. To make a reservation, RSU will call the smart contract function which incurs some gas cost therefore, before making the reservation, EV needs to pay RSU to make a reservation along with a valid signature. Although ring signatures provide privacy to the EV's identity, we propose a mixing scheme, e.g., tornado cash[1] for EVs to pay RSUs for reservations. The mixing service helps the user to mix their cryptocurrency using specially developed algorithms to secure the sender's identity. This will provide an extra layer of anonymity to EVs while paying RSUs.

[1] https://app.tornado.cash/

Algorithm 5.1 Ring signature generation

**Input:** G, M, $sk_S$
**Output:** A ring signature $\sigma = (pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n)$

Initialize $G$, $x_i$, $v$, $M$, $y_i \leftarrow 0$
$EV_S$ selects CS and sets charging information as $M$ (Danish *et al.*, 2020c)
*Step 1:* Select public keys $G = \{pk_1, pk_2, ..., pk_n\}$
*Step 2:* Calculate the key $k = h(M||(pk_1, pk_2, ..., pk_n))$
*Step 3:* $EV_S$ generates a random value $v \in \{0, 1\}^b$
*Step 4:* $EV_S$ generates *n-1* random value $x_i \in \{0, 1\}^b$
$\{y_1, ., y_{s-1}, y_{s+1}, ., y_n\} = E_{pk_i}\{x_1, ., x_{s-1}, x_{s+1}, ., x_n\}$
*Step 5:* $EV_S$ solves the combining function to find $y_s$
$C_{k,v}(y_1, ..., y_s, ..., y_n) = E_k(y_n \oplus (...E_k(y_s \oplus (...E_k(y_1 \oplus v))) = v$
*Step 6:* $EV_S$ calculates $x_s$ from $y_s$ using its private key $sk_S$, $x_s = Decrypt_{sk_S}(y_s)$
*Step 7:* $EV_S$ will send the ring signature
$\sigma = \{pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n\}$ along with $M$ to the corresponding RSU

The smart contract reservation function will first compare the reservation information taken as input from CS and RSU before confirming and sending a transaction to the blockchain network. This is to ensure that CS has not changed the charging parameters provided at the time of selection by EVs, e.g., the selected time slot, and charging price (Danish *et al.*, 2020c). If the required information matches, then the function will be executed and a transaction will be sent to the blockchain network, thus, the reservation will be written on an immutable distributed ledger. Finally, once the reservation is made, the RSU will broadcast this occupied charging slot information to all the EVs in the vicinity.

### 5.3.3 Ev authentication

After the charging slot is reserved at CS and information is stored on the blockchain, the EV owner needs to reach the CS premises to charge the vehicle at the reserved time. As any EV can reach CS at the said time to claim a charging slot, EV needs successful authentication by the CS before starting vehicle charging. We propose a public key-based EV authentication mechanism for the CS, where an EV can generate a fresh public key for authentication purposes and share

it with the RSU along with message $M$. Once the RSU has made the reservation, the CS will encrypt the corresponding blockchain transaction hash along with a freshly generated random number with the EV's public key and share it with the corresponding EV. The transaction hash corresponds to the charging slot time. Upon receiving, the EV will decrypt it using its private key and subsequently, it will show this information to the CS at the CS premises to authenticate itself to start charging. It must be noted that the public key is freshly generated for one time use, and it will not leak any private information about EV.

The idea of using a transaction hash along with a random number helps in reservation information verification along with EV authentication. When CS encrypts this information with the EV's public key, the EV can easily verify the validity of reservation by looking at the content of the successful transaction, i.e., charging slot time and charging price. Moreover, a random number is necessary to ensure the validity of authentication instead of just sending a transaction hash. This is because RSU cannot be trusted and this transaction hash can be leaked intentionally or unintentionally to other entities including EVs and any EV can show transaction hash to CS at its premises. Therefore, a freshly generated random number is equally important for EV authentication.

### 5.3.4 Fair payment in trust-less environment

This subsection addresses the following problem: *how can smart contract-based payments can be executed between EVs and CSs while ensuring the unlinkability of an EV's blockchain address with its physical identity?*

According to current blockchain-based EV charging systems Danish *et al.* (2020c); Xu, Chen & He (2021); Baza & et al (2021), EVs arrive at CS premises and charge themselves. When the EV is fully charged, it calls the smart contract function to pay CS for its charging services and departs. Current designs have the problem that the CS can view the sender address of transactions on the blockchain network, and in this way, the CS can link the physical identity of EVs with the

blockchain address. With the blockchain address exposed, CS can see the charging history and payment history of the EV, which is dangerous for privacy.

Algorithm 5.2 Ring signature verification

**Input:** M, $\sigma = (pk_1, pk_2, ..., pk_n; v; x_1, x_2, ..., x_n)$
**Output:** *{0,1}*

Initialize $G$, $x_i$, $v$, $M$, $y_i \leftarrow 0$
*Step 1:* RSU computes key $k = h(M||(pk_1, pk_2, ..., pk_n))$
*Step 2:* RSU calculates $y_i$ from $x_i$ using public keys
$\{y_1, ., y_{s-1}, y_{s+1}, ., y_n\} = E_{pk_i}\{x_1, ., x_{s-1}, x_{s+1}, ., x_n\}$
*Step 3:* RSU checks the correctness of the equation
$C_{k,v}(y_1, y_2, ..., y_n) = E_k(y_n \oplus (...E_k(y_s \oplus (...E_k(y_1 \oplus v))) = \mathrm{v}$
*Step 4:* If step 3 holds, RSU reserves charging slot at CS

As a solution, we propose an ether-mixer smart contract that takes one-time deposits from EVs (e.g., at the start of every month) and aggregates them to ensure that blockchain addresses are not linked to deposits. Additionally, to execute the payment functionality for the charging services, the smart contract keeps track of each CS's balance and then adds it to the address associated with that CS. Furthermore, the smart contract will lock the amount of each CS for a certain period of time, e.g., a day. The total amount transferred to CS at the end of the day is not distinguishable from where the individual amounts originated. To ensure normal protocol execution among EVs and CSs, we also propose a deposit from CSs. The deposit will also ensure that conflict resolution protocols are followed properly, and a deviation from normal behavior will incur a penalty paid to the honest party. To begin operations, the proposed smart contract will be deployed by the RSU. Smart contracts have different functionalities such as storing funds, calculating payments, and transferring funds. In addition, the contract will inherit the reservation contract to access the reservation parameters. The deposit functionality of the smart contract will work in such a way that it will allow EVs to deposit funds with the help of the *deposit()* function, where EVs will transfer the chosen amount to the smart contract address. The function will assign the deposited amount to the user's EV balance and the total balance of the smart contract will add up after every deposit. Every EV user's balance will be tracked by an

Algorithm 5.3 Fair payment smart contract design

**Contract Deployment:**
RSU deploys the contract on blockchain network
RSU broadcasts the contract address
Initialize $releaseTime$, $mapping(address => balance)$ balances
***function* Deposit(amount, contract address)**
$EV \rightarrow transfer(amount, contractaddress)$
$balance = BalanceOf(msg.sender)$
$balances[msg.sender] = balance + amount$
***function* BalanceOf(address)**
return $balance = balances[msg.sender]$
***function* WithdrawEV()**
require $(msg.sender == registeredEV)$
$contract \rightarrow transfer(amount, EVaddress)$
$balances[msg.sender] = 0$
***function* WithdrawCS()**
require $(msg.sender == registeredCS)$
require $(releaseTime > now)$
$contract \rightarrow transfer(amount, CSaddress)$
$balances[msg.sender] = 0$
***function* Payment(amount)** $balances[EV] -= amount$
$balances[CS] += amount$

array mapping of addresses and balances. Since these deposits will be made at the inception of EV's charging service, one cannot link the physical identity with a certain address.

Additionally, the same deposit amount will be used to pay for CS charging services. The RSU will sign a random transaction hash with EV's public key and share it with the EV as deposit proof. When EV makes a reservation at CS, it sends the transaction hash as proof of deposit. CSs will also use the deposit function to deposit some money as a guarantee that they will execute the protocol correctly. Penalties will be deducted for malicious behavior by CSs. The smart contract also defines three other functions related to EV and CS deposits, i.e., *BalanceOf()*, *WithdrawEV()* and *WithdrawCS()*. The *BalanceOf()* function will help EVs obtain their remaining balance and will also help RSUs to check if EVs have sufficient balance to reserve a time slot at CS. The

*WithdrawEV()* and *WithdrawCS()* functions will allow EVs and CSs to withdraw the deposited amount back to their own account, respectively.

After the EV reaches the CS and charges itself, it must pay the CS to use the charging services. RSU is notified through an offline channel that the charging has been completed based on certain parameters. As a result, the RSU will verify the reservation smart contract parameters and confirm with the CS. If both parties agree, the RSU will call the *payment()* function of the proposed smart contract. The amount will be deducted from the account of EV and added to the account of CS in this function. The amount will not be deposited into the CS account at this time as RSU will lock CS's balance for a certain period of time, e.g., one day. RSU will keep following the same protocol for other EVs and add the amount to CS's balance. At the end of the day, RSU will call the *withdrawCS()* function to transfer the overall amount to CS's address. The *withdrawCS()* function can be called either by RSU or CSs themselves. The smart contract functions are shown in the Algorithm. 5.3.

It should be noted that at the end of the day, CS does not know what amount pertains to which EV once it receives payment for its charging services. The smart contract will aggregate the amounts in such a way that no CS can link the amount with its associated blockchain address. Because the CS is unaware of the blockchain address, it cannot link the blockchain address to the EV's physical identity at the CS premises. EVs communicate with RSUs through ring signatures and RSUs cannot link the physical identity of EVs with their blockchain address, therefore preserving end-to-end privacy for the EV owner.

## 5.4 CS information verification mechanism

In this section, we describe a mechanism for verifying the availability of charging slots at CSs. In the protocol, CSs broadcast their charging slot availability information to EVs at the beginning, which is used to select the best CS. Because EVs must trust the available information provided by CSs to make a reservation, and there is no way to verify the validity of the information, there is a possibility of a reservation conflict. In this work, we use homomorphic encryption, where

the encrypted reservation information can be added up for each CS ID to obtain the total number of charging slots available at each CS without having to know the actual reservation information of each EV.

Algorithm 5.4 CS information verification

Initialize $G$, $H$, $p$, $q$, $g$, $r$
**Key Generation:**
$EV_s$ generates $p$ and $q$ and calculates $n = pq$
$EV_s$ calculates $\lambda = lcm(p-1, q-1)$ and $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$
$EV_s$ calculates public key $(n, g)$ and private key $(\lambda, \mu)$
**Secure summation:**
$EV_s$ encrypts message, $[x_s] = E(x_s) = g^{x_s}.r^n \bmod n^2$ and sends it to next $EV_1$
$EV_1$ encrypts $[x_1] = g^{x_1}.r^n \bmod n^2$, perform product of $[x_s].[x_1] = [x_s + x_1]$, and forwards to next $EV_2$
$N^{th}$ EV forwards $[x_s + x_1 + x_2,..., x_{n-1}].[x_n]$ to $EV_s$
**Summation Result:**
$EV_s$ decrypts final message with private key $(\lambda, \mu)$
$Sum = L([x_s + x_1 + x_2, ..., x_{n-1}, x_n]^{\lambda} \bmod n^2).\mu \bmod n$
Repeat for other CSs and broadcast result to other EVs

One can argue that anonymity is already coming through the ring signature-based scheme, however, CS information verification is a separate mechanism, and is not related to privacy-preserving reservation. Sharing the reservation information with other EVs or a central entity without encryption could lead to EV's privacy leak and will invalidate the ring signature based reservation mechanism, as everyone can see which EV has made reservation to which CS.

**Set-up:** Let $G = (EV_1, EV_2, ..., EV_i)$ be the list of EVs who made a reservation at one of the CS in $H$, where $H = (CS_1, CS_2, ..., CS_j)$. Let $x \in \{0, 1\}$ denote if a vehicle $EV_i$ has made a reservation at any $CS_j$. To execute the protocol, an EV will be selected randomly from the list of EVs to generate a public/private key pair using the Paillier crypto-system and is denoted as $EV_s$. It is assumed that all the EVs taking part in the protocol are honest. Moreover, the protocol will run for one CS at a time. At the basic level, the verification algorithm will be executed as in the Algorithm. 5.4. More details about the protocol are given below.

### 5.4.1 Key generation

The $EV_s$ will first generate the public/private key pair using the Paillier crypto-system by generating two large prime numbers $p$ and $q$ independently and randomly. Let $n = pq$ and $\lambda = lcm(p-1, q-1)$. It will then choose a random number $g \in \mathbb{Z}^*_{n^2}$, while ensuring that n divides the order of $g$ by checking $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, where $L(y) = (y-1)/n$. After calculating the values of $g$ and $\mu$, the public and private keys are defined as $(n, g)$ and $(\lambda, \mu)$, respectively. This public key is broadcast to all the other EVs, while EV keeps the private key to itself.

### 5.4.2 Secure summation

$EV_s$ will start the protocol by generating a random number $r \in \mathbb{Z}^*_n$ and encrypting its own message $x$ with the public key, i.e., $[x_s] = E(x_s) = g^{x_s}.r^n \bmod n^2$, where $E()$ represents encryption function and $[x_s]$ represents encrypted message. After encryption, $EV_s$ will send the encrypted message to the next $EV_1$. $EV_1$ will encrypt its own message $x$ with the public key, i.e., $[x_1] = g^{x_1}.r^n \bmod n^2$. After encryption, it will perform product of $[x_s].[x_1]$, which yield $[x_s + x_1]$. Once it is calculated, $EV_1$ will send the encrypted sum to the next EV, i.e., $EV_2$ and finally, the encrypted message $[x_s + x_1 + x_2, \ldots, x_{n-1}]$ will reach the final $EV_n$.

### 5.4.3 Summation result

$EV_1$ will encrypt its own message $x$ with the public key, i.e., $[x_n] = g^{x_n}.r^n \bmod n^2$ and send the product $[x_s + x_1 + x_2, \ldots, x_{n-1}].[x_n]$ back to $EV_s$. $EV_s$ will decrypt the message using the private key pair $(\lambda, \mu)$, i.e., $sum = L([x_s + x_1 + x_2, \ldots, x_{n-1}, x_n]^\lambda \bmod n^2).\mu \bmod n$. This protocol is repeated until all the reservation slots are calculated for each CS. After calculation, the $EV_s$ broadcast this information to all other EVs and all the EVs can cross-verify the charging information provided by CSs.

Following this protocol, the $EV_s$ is able to decrypt the final message to obtain the summation without knowing the actual value of any EV. Therefore, privacy-preserving secure summation can be achieved.

## 5.5 Conflict resolution mechanism

In this section, we discuss a conflict resolution mechanism that can be used to resolve disputes between EVs and CSs, while penalizing abnormal behavior. Due to the lack of conflict resolution mechanisms for EVs and CSs in current works and their inability to prevent disputes, it is necessary to enable dispute resolution among EVs and CSs without the involvement of trusted third parties. The solution to this problem is to use the hardware root of trust and TEEs, where the program inside the hardware is virtually a black box that cannot be altered or observed by an attacker or device (Al-Bassam *et al.*, 2018).

We assume that each EV and CS has a smart meter, with a trusted computing base (TCB), where the hardware root of a trust resides. Only the output generated from this TCB is trusted in the EV's and CS's smart meters. These outputs from EVs and CSs are communicated with RSUs using a secure link. This secure hardware runs the energy-related programs, i.e., how much energy has been consumed by the CS and EV. This information is then directly fed to the conflict resolution smart contract with the help of RSU. Based on the input, a smart contract calculates and imposes a penalty on an abnormal party. Moreover, the authenticity and integrity of this information are also checked periodically with the help of remote attestation, which guarantees the security of the device. This remote attestation is performed by a security service using standard token verification. If the remote attestation verification is successful, the token is generated and signed with the security service's secret key, which represents proof that the information generated using the secure hardware can be trusted. Examples of such services are Intel attestation services and Azure sphere security.

At the CS premises, charging services are provided to the EV. In the normal execution of the protocol, the EV will charge itself and pay the CS for the charging services. However, there

is a possibility that either EVs or CSs do not execute the protocol normally. In this case, the normal party will notify the RSU about the possible conflict through an offline channel. A smart meter in the EV normally consists of an application and a cryptographic component. The cryptographic component in the smart meter creates the attestation tokens using the key stored in the hardware root of trust, which is periodically sent to the security service to generate the attestation proofs. To start the conflict resolution protocol, EV or CS will execute a program in the TEE, which calculates the energy consumption over a period of time. The output of this program along with the attestation proof will be sent to the RSU. Upon receiving the EV and CS energy consumption information update from a smart meter, the RSU verifies the message by checking for a valid signature and remote attestation proof in the message. After verification, the RSU will call the conflict resolution smart contract function with the input generated by EV or CS. Examples of programs running in the TEE can be used to calculate the state of charge of an EV's battery over a period of time, and the energy consumed by CS over a period of time.

The smart contract will take the input from the RSU and check if CS has provided the charging services correctly or if EV is fully charged as per the reservation information. After verification, the smart contract will deduct the balance from the abnormal party and transfer it to the normal party. This contract will also inherit the reservation contract to access the charging parameters stored at the time of reservation. These parameters will be used to compare the reservation information with the information generated by the TEE program. The smart contract function will be called by RSU similar to the reservation function to ensure end-to-end privacy for EV users and to ensure the unlinkability of public blockchain addresses with EVs' physical identity.

However, it must be noted that TEE is vulnerable to different security threats and one must not rely 100% on TEE for trusted execution of the conflict resolution mechanism. Some common attack patterns against TEE are physical attacks, privileged software attacks, software attacks on peripherals, address translation attacks, cache timing attacks, and side channel attacks, etc. In future, we plan to explore other privacy-preserving methodologies to implement the conflict resolution mechanism.

## 5.6 Security analysis

This section presents the security and privacy analysis of the proposed protocol. We first discuss how the proposed protocol achieves the security requirements explained in Section. 5.2 followed by a detailed analysis on resilience against multiple security attacks.

Algorithm 5.5 Conflict resolution mechanism

1 RSU deploys conflict resolution smart contract
2 $EV$ or $CS$ notifies RSU about possible conflict
3 $EV$ or $CS$ executes the program in TEE to measure energy consumption
4 RSU receives and verifies the digitally signed output and remote attestation
5 RSU provides input to the smart contract
6 Smart contracts penalize abnormal parties and balance is transferred to normal parties

### 5.6.1 Security properties

**EV Privacy:** The main objective of this work is to ensure EVs' privacy at the time of charging slot reservation and payment, i.e., no entity in the network knows the real ID of EVs or their associated blockchain addresses.

**Theorem 1.** The proposed protocol ensures EV anonymity against RSUs and CSs.

*Proof:* In our proposed protocol, EV is communicating with RSUs with the help of ring signatures, i.e., by signing message with multiple public keys as $\sigma = (pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n)$. The ring signature guarantees the anonymity of EV as the message is signed by many public keys and the probability of finding the author of the ring signature message is approximately $\frac{1}{n}$. If k private keys are exposed out of n public keys, this probability increases to $\frac{1}{n-k}$. As the anonymity guarantee is directly proportional to the number of public keys used to sign the message, more public keys can be used to decrease the probability in the case of exposed private keys. Similarly, RSU is making a remote reservation at CS on behalf of EV by using a smart contract. The privacy of EVs' blockchain address will be preserved and unlike previous work (Danish *et al.*, 2020c), it will not be revealed to a selected CS even after reaching

the CS premises. Thus, no entity in the network including the RSU and CS knows about the EV's identity inside or outside the blockchain, during or after charging slot reservation.

**Identity Unlinkability:** Compared to previous works, this work ensures the EV identity unlinkability property, i.e., dissociation of the EV physical ID with the blockchain address. First, the EV uses ring signatures to communicate with RSUs to hide their identity. Moreover, payment to the RSU is done through crypto-mixing services, which provides additional anonymity to EV. Finally, the RSU makes a remote reservation on the behalf of the EV, which ensures the unlinkability of the EV's ID with its blockchain address. Even if the CS colludes with the RSU and shares the EV's physical identity with RSU at its premises, the RSU cannot determine the blockchain address of the EV because of ring signatures communication, thus ensuring identity unlinkability property of EVs.

Moreover, one can also consider a scenario, where the security camera at the CS premises tries to record the car's activity based on the plate number. Again, similar to the physical identity of the EV owner, plate number cannot be associated with the blockchain address. One can keep track of when the car comes to the CS, or where it goes and try to learn the car's activity, but this comes under the category of physical security and is out of the scope of this work. The aim of this work is to ensure the unlinkability of any physical asset with the blockchain address, and based on the proposed scheme, one cannot link the plate number to the blockchain address, thus preserving the privacy of EV owner. However, in case of small number of EVs coming to the CS on a certain day or a period of days, one can see the balances of the addresses in the smart contract transaction history, which got changed over this period of time. A statistical correlation analysis of physical identities/plate number with the blockchain address can compromise the EV owner's privacy. We plan to analyze and solve this problem in future work.

Another thing to be noted is that ensuring the identity unlinkability will create the problem in traceability. As we cannot link the blockchain address with the physical identity, we cannot trace back the EV. To solve this problem, a government authority can be introduced which keeps track of the identity and blockchain information in case something goes wrong.

**Authentication:** The proposed scheme ensures secure EV authentication at CS premises. At the time of reservation, CS generates a random number and encrypts it with the EV's public key along with the transaction hash as,

$$CS \rightarrow EV : input = \left\{Enc_{pk_{EV}}(txnhash, rand)\right\}.$$

The transaction hash contains the reserved charging slot information for a given EV. At the CS premises, EV needs to decrypt and share the message with its private key, which is matched by CS for authentication purposes as,

$$EV \rightarrow CS : \left\{Dec_{sk_{EV}}(input)\right\}.$$

$$CS_{compare} : \{(txnhash, rand)_{EV} || (txnhash, rand)_{CS}\}.$$

As this encrypted message can only be decrypted with the private key possessed by EV, it cannot be decrypted by malicious entities in the network, thus leading to secure authentication.

**Data integrity:** The reservation information related to EV is stored on the blockchain network. As a blockchain is an immutable ledger, no entity in the network can tamper with the reservation information, i.e., charging price, charging slot time, etc., thus, ensuring data integrity and immutability. Moreover, as the information is immutable, it will be helpful in the auditability of the reservation information.

**No trusted intermediary:** Unlike traditional centralized EV charging networks, in our protocol, a smart contract-based blockchain network is employed for charging slot reservations. EVs do not need to trust a centralized third party to schedule a charging slot. Moreover, as the reservation information is stored in the blockchain network, i.e., immutable ledger, it is possible to trace any reservation in case of a possible issue, thus ensuring traceability.

However, it must be noted that it is possible for two EVs to reserve the same charging time slot at the CS. Considering the fact that RSU makes a reservation for EV at the CS, it is possible for

the RSU to reserve same charging slot without having the knowledge of EVs. We plan to solve this problem in the future work.

### 5.6.2 Security attacks analysis and comparison

**Tamper Attack:** Once the reservation information is stored on the blockchain network, it cannot be changed or modified due to the tamper-proof feature of blockchain. Moreover, the ring signatures cannot be modified or created by a malicious entity. A malicious entity will generate $\sigma' = \{pk_1, pk_2, ..., pk_s, ..., pk_n; v; x_1, x_2, ..., x_s, ..., x_n; M\}$, which will result in a key $k' = h(M||(pk_1, pk_2, ..., pk_n))$. Ring verification will fail if the this key is used to verify the signatures, i.e., $C_{k',v}(y_1, y_2, ..., y_n) = E_{k'}(y_n \oplus (...E_k(y_s \oplus (...E_k(y_1 \oplus v))) \neq$ v. Similar reasoning can be provided for false data injection attacks. Therefore, the proposed protocol provides protection against tamper and false data injection attacks, thus, preserving data integrity.

**Repudiation Attack:** In our proposed protocol, the smart contract reservation function is executed only once the charging parameter information obtained from EV and CS matches. Once the smart contract executes the reservation function, the information will be written in the blockchain network. Since each entity has a specific blockchain address and transactions are signed by public keys, no entity in the network can repudiate the information once it is stored on the blockchain network, thus the protocol protects against repudiation attacks.

**DoS Attack:** A malicious EV can flood RSUs with charging reservation requests, however, the proposed protocol protects against these kinds of attacks as a legitimate EV user communicates with the RSU by using a ring signature and it needs to provide payment proof before the reservation information is stored on the blockchain network. The RSU will not execute the smart contract function until it obtains the proof of payment from an EV. Thus, the payment functionality before reservation mitigates possible DoS attacks.

**Man-in-the-middle (MITM) Attack:** In the proposed protocol, EVs communicate with RSUs by using ring signatures. Based on the correctness and soundness properties of ring signatures, an honest RSU verifier will always accept a valid ring signature and if someone tries

Table 5.1 Security comparison

| Functionality | Proposed | (Danish *et al.*, 2020c) | OCPP | ISO15118 | [6] | (Said *et al.*, 2013) | (Yang *et al.*, 2013) | (Bodet *et al.*, 2012) |
|---|---|---|---|---|---|---|---|---|
| No trusted intermediary | ✓ | ✓ | *x* | *x* | ✓ | ✓ | ✓ | *x* |
| End-to-End privacy preservation | ✓ | *x* | *x* | *x* | *x* | *x* | *x* | *x* |
| Identity unlinkability | ✓ | *x* | *x* | *x* | *x* | *x* | *x* | *x* |
| Traceability & Accountability | ✓ | ✓ | *x* | *x* | *x* | *x* | ✓ | *x* |
| Data Integrity | ✓ | ✓ | *x* | *x* | ✓ | ✓ | ✓ | ✓ |
| CS Information Verification | ✓ | *x* | *x* | *x* | *x* | *x* | *x* | *x* |

to change/modify the ring signature or message on the way, the RSU will reject it as a false one. Thus, our design can oppose the MITM attack.

**Fake Reservation Attack:** A malicious EV can make a fake charging reservation at CS and never appear. As explained before, the proposed protocol requires payment proof from EVs to RSUs before a charging request is written on the blockchain network. Therefore, this functionality of funds deposited before each reservation makes it hard for a malicious EV to launch a fake reservation attack.

Table. 5.1 provides a detailed comparison of the security properties of our proposed protocol with state-of-the-art schemes. It must be noted that, unlike previous frameworks, our proposed protocol offers end-to-end privacy preservation for EV, where the physical identity of EV cannot be linked with its blockchain address.

## 5.7 Performance evaluation

This section evaluates the proposed protocol in terms of performance along with blockchain transaction and storage overhead.

### 5.7.1 System setup

We run our experiments on an $8^{th}$ Generation Intel $Core^{TM}$ i7-8650U Processor with 8 GB RAM. We have implemented ring signatures and the proposed authentication protocol in the Python programming language to validate the effectiveness and feasibility of the proposed protocol. We implemented the blockchain functionality through the Go language implementation

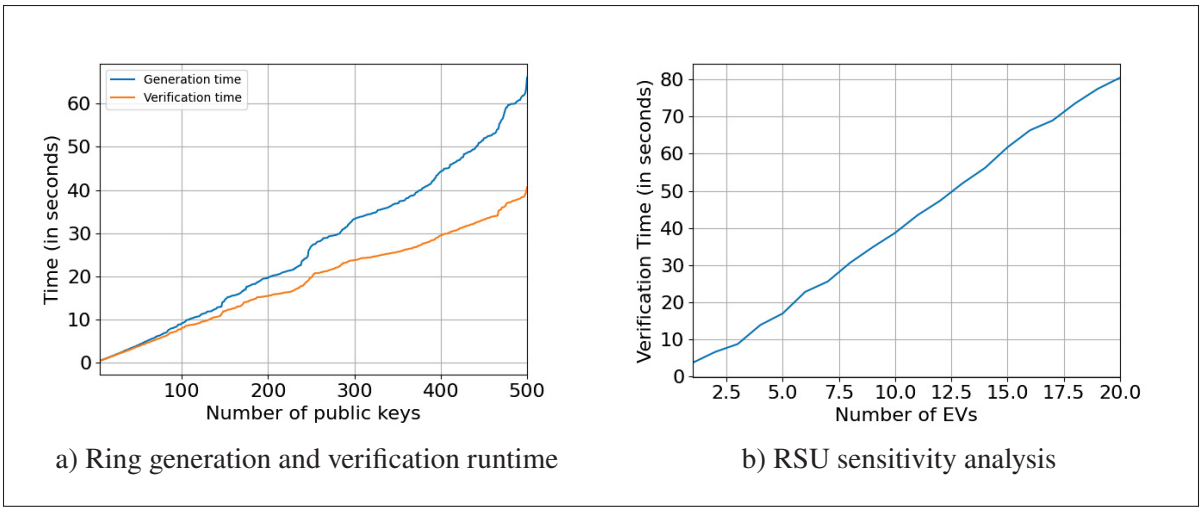
(Geth) and web3py, a Python library to interact with the Ethereum network. Moreover, we use the Solidity programming language to implement the smart contract.

### 5.7.2 Results

**Computation time:** We first investigate the performance of the proposed ring signature-based protocol in terms of ring generation and verification runtime as shown in Fig. 5.3a. These experiments are run multiple times to average out the runtime values with a different number of public keys. Moreover, we use the ECC based ECDSA python module to generate public and private keys. Fig. 5.3a shows that increasing the number of public keys increases the generation and verification time of the ring signature at the RSU. Here the public keys corresponds to the number of EVs, one public key per EV. For a large number of public keys, it can take up to 1 minute to generate a ring. However, the time seems reasonable assuming EV needs to reserve a time slot once in a day. The generation time depends on the computation power, and there exists a trade-off between the number of public keys and EV user's privacy.

An EV can use a large number of public keys to generate a ring to ensure high anonymity. One can reduce the computation time by using fewer public keys, however, it comes at a cost of low anonymity. Using the large number of public keys to generate ring signature give rise to a limitation. A statistical analysis can find a good compromise between the security and computation time trade-off, which is one part of the future work. It is possible that two EVs want to reserve the same time slot at the same time, which could result in possible conflict. We plan to resolve this problem in future work.

We then perform RSU sensitivity analysis, i.e., the ring verification time with an increasing number of EVs as shown in Fig. 5.3b. To perform this experiment, we generate an ECC-based public/private key pair randomly for each EV to generate the ring signatures. We then calculate the ring verification time with an increasing number of EVs while considering one RSU. Fig. 5.3b shows that increasing the number of EVs increases the overall verification time. Considering the first in first out (FIFO) scenario, EVs need to wait in a queue to communicate with the RSU

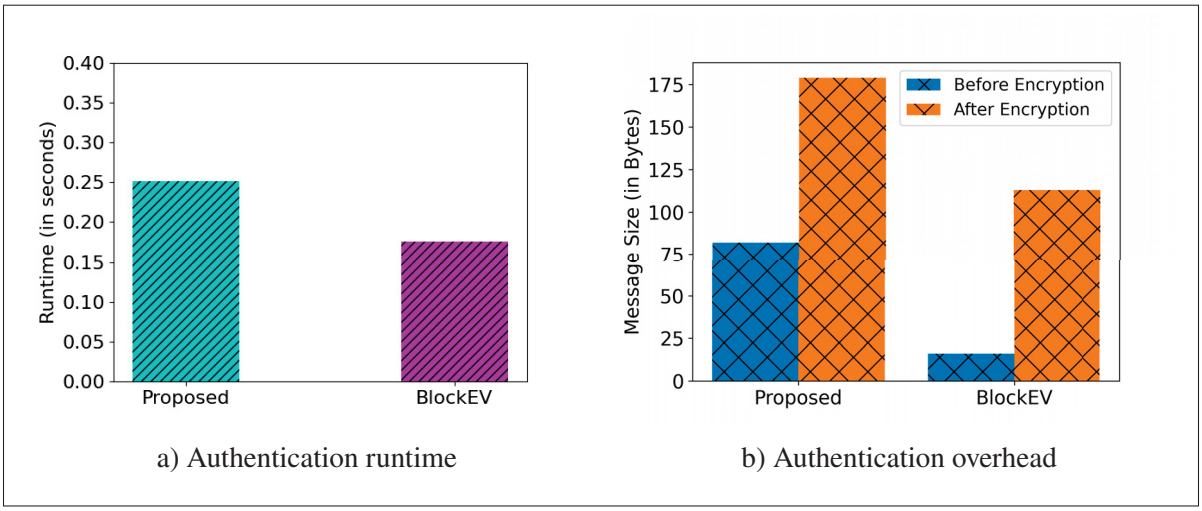



a) Ring generation and verification runtime

b) RSU sensitivity analysis

Figure 5.3 Ring signatures analysis

to reserve a slot. However, parallel execution at the RSU server can further decrease the time. Moreover, the waiting time also depends on the number of public keys used by each EV.




a) Authentication runtime

b) Authentication overhead

Figure 5.4 Authentication analysis

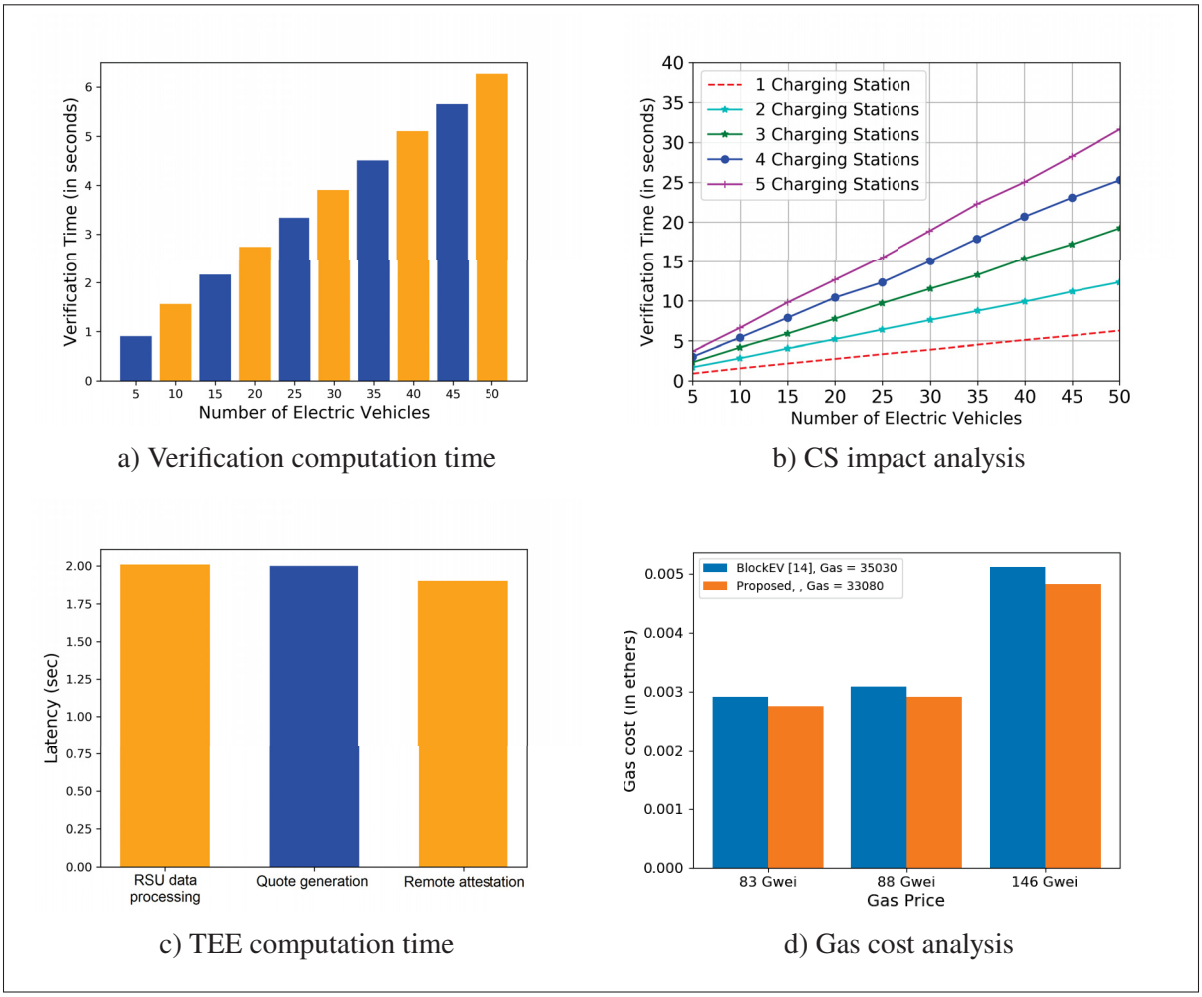
It may also be noted that the experiments are performed on laptop however, in real-world scenarios, the EV owners will use their mobile phones or in-vehicle infotainment to make the charging reservations. The computation capability differs among these devices.

**Authentication analysis:** We further analyze the proposed authentication method in terms of computation time and overhead and compare it with the state-of-the-art methods. We generate a set of ECC-based public/private keys to perform this experiment. Moreover, we ran this experiment 1000 times to average out the result as shown in Fig. 5.4a and Fig. 5.4b. Fig. 5.4a shows the computation time of the proposed authentication protocol and compares it with a state-of-the-art solution. The runtime includes the encryption time of the message by EV and decryption time by the selected CS to verify the authenticity. It can be seen in Fig. 5.4a that the proposed protocol incurs more time to authenticate the EV compared to the work in (Danish *et al.*, 2020c). This is because, in the (Danish *et al.*, 2020c) solution, EV communicates directly with CS however, in our work, the EV communicates with the RSU, which incurs more time in the exchange of messages during the authentication process. However, the authentication time seems reasonable assuming EV needs to authenticate only once at the CS premises at the time of charging.

Similarly, Fig. 5.4a shows the computation time of the proposed authentication protocol and compares it with a state-of-the-art solution. The message size and encrypted message size are high compared to the state-of-the-art (Danish *et al.*, 2020c) solution. This is because, in the (Danish *et al.*, 2020c) solution, only a random number is used to authenticate EV. However, we use a transaction hash along with a random number to authenticate EV at the selected CS in our work. Although our proposed protocol incurs more computation time and storage overhead, unlike the (Danish *et al.*, 2020c) solution, our proposed authentication method also allows EVs to validate the reservation information provided by the RSU at the time of reserving a time slot at CS. Finally, considering the implementation of the protocol as a mobile application, the increased authentication time and overhead are still reasonable.

**Blockchain overhead:** We now investigate the blockchain storage and transaction overhead of the proposed protocol in Fig. 5.6b and Fig. 5.6a, respectively. Fig. 5.6a shows the number of transactions incurred by the proposed protocol and compares it with a state-of-the-art solution. Since different schemes depend on the number of EVs and CSs, we consider $N = 1000$ EVs and $M = 5$ CS to ensure common ground for comparison. Our proposed protocol incurs 2

transactions, i.e., reservation and payment, therefore for $\mathcal{G}$ number of EVs, the total number of transactions will be $2\mathcal{G}$ compared to $4\mathcal{G}$ and $2M + NMn$ for the (Danish *et al.*, 2020c) solution and method presented in [9], respectively, where M, N, and n are the numbers of EVs, CSs, and bids respectively. It can be seen from the plot that compared to the state-of-the-art methods, our proposed protocol incurs less transaction overhead. Unlike the (Danish *et al.*, 2020c) solution, our design performs authentication off-chain, thus, leading to less transaction overhead. Moreover, compared to (Knirsch *et al.*, 2018), our proposed solution does not include bids from CSs.




a) Verification computation time

b) CS impact analysis

c) TEE computation time

d) Gas cost analysis

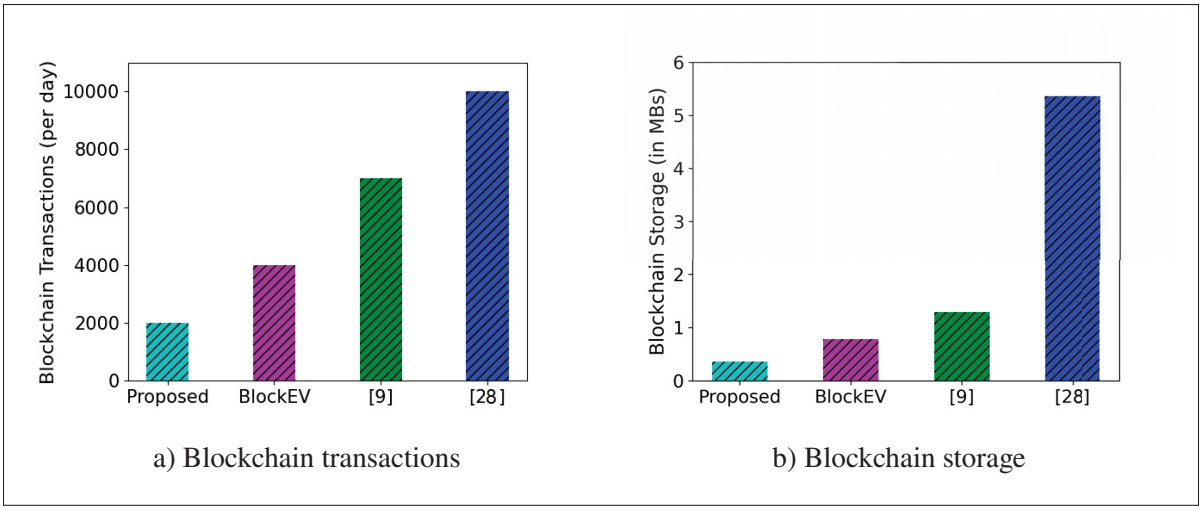
Figure 5.5 CS information verification analysis

Similarly, blockchain storage overhead analysis can be seen in Fig. 5.6b. Our storage overhead analysis is similar to the work in BlockEV (Danish *et al.*, 2020c). The only difference is that we are not using any reputation or rating in our proposed protocol, which leads to low storage overhead in our proposed protocol. Moreover, compared to the (Danish *et al.*, 2020c) solution, our proposed protocol incurs less transaction overhead, which leads to less storage overhead. Fig. 5.6b shows that the proposed protocol incurs less storage overhead compared than BlockEV, (Knirsch *et al.*, 2018) and (Gao *et al.*, 2018).

**CS information verification analysis:** We further investigate the performance of the proposed CS information verification system. To perform the experiments, we use PHE, a Python-based library to generate the keys through the Paillier cryptosystem, and to encrypt and decrypt the EV's reservation information. Moreover, we ran these experiments several times to average out the results as shown in Fig. 5.5a and Fig. 5.5b. This verification time includes the key generation, encryption and decryption time. Additionally, the experiments are performed with localhost, therefore the delay in the network can change the overall verification time.

Fig. 5.5a shows the overall time to verify the CS information with an increasing number of EVs for one CS. Fig. 5.5a shows that as the number of vehicles increases, the verification time increases. An increasing number of EVs will result in more encryption operations before the information can be decrypted, thus increasing the verification time. Similarly, Fig. 5.5b shows the verification time with varying numbers of charging stations. It can be seen in the figure that as the number of CS increases, the verification time increases. This is because the protocol explained in algorithm 5.4 executes individually for each CS. This result shows the worst-case scenario for multiple CS verification however, this time can be improved if EVs execute the protocol in parallel for multiple CSs.

**Conflict resolution analysis:** We further investigate the performance of the TEE based conflict resolution mechanism in terms of computation time. To perform the experiments, we assume an Intel based TEE is installed on EV's smart meter and communicates with the RSU server. The RSU server is further connected to the blockchain network to provide input to the conflict




a) Blockchain transactions b) Blockchain storage

Figure 5.6 Blockchain analysis

resolution smart contract. The experiments were executed on an Ubuntu 16.04.7 with a quad-core Intel i7-8665U @1.9GHz processor and 8GB RAM. To simulate trusted execution environment we implement Docker version 20.10.7 build f0df350 and SCONE along with its public CAS scone-cas.cf version 5.6.0. Moreover, we run the experiments several times to average out the results.

First the EV or CS will communicate with RSU on an offline channel about the possible dispute. The RSU will then call the execution function on the EV's or CS's smart meter through the web API as shown in Fig. 5.5c for RSU data processing. It takes approximately 2.01 seconds for the enclave to process it because of network shielding, which allows network connections to terminate inside the enclave. Once the connection is established, the function is run in the enclave and a quote is generated for attestation of approximately 2 seconds. Finally, the RSU performs remote attestation via an Intel Attestation Service (IAS) and this verification takes approximately 1.901 seconds. Once remote attestation is performed, the RSU calls the smart contract function to handle dispute. The TEE execution and remote attestation time are reasonable, assuming a rare event of dispute.

**Smart contract analysis:** We provide a gas cost estimation to evaluate the smart contract functionality of the proposed protocol and compare it with the state-of-the-art methods. The

gas cost is a unit that measures the amount of computational effort that a smart contract will take to execute certain operations. In our proposed protocol, a smart contract function named *reservation* is called by RSU on behalf of EV to reserve a charging slot at the CS and write it on the blockchain network. Fig. 5.5d shows comparison of the proposed reservation smart contract function with the work in (Danish *et al.*, 2020c). It can be seen in the table that the required gas amount to execute the reservation function of the proposed scheme is lower than that in (Danish *et al.*, 2020c). It should be noted that the confirmation time of the transaction depends on the gas price. Increasing the gas price results in low confirmation time, however, it comes at the cost of a high transaction fee. We have also provided the gas cost with different gas prices, i.e., 83, 88, and 146 Gwei, for cheap, average, and fast transactions, respectively. The average confirmation time of a transaction in the blockchain decreases significantly by increasing the Gwei gas price. We also provide the gas cost analysis of the payment functionality in Table 4.2. The average gas price, i.e., 88 Gwei is selected to calculate the overall gas price, however, to increase the blockchain confirmation time, one can use a high gas price in Gwei.

Considering an average gas price of 88 Gwei, it takes 0.002911 ethers (almost 11 dollars, where 1 ETH = $3800) to reserve a charging slot on the blockchain network. In case of increased Ethereum price, the price to reserve a charging slot will increase significantly. To decrease the overall gas cost, one can use Layer 2 blockchain solutions, e.g., Loopring, which reduces the gas fees to 1/30 - 1/100 of the gas fees on the Ethereum mainnet. Finally, any public blockchain with cheap gas fee can be incorporated in our proposed design.

Table 5.2 Blockchain transaction and storage comparison

| Function | Gas cost | Gas price (in Gwei) | Gas price (in Ethers) |
|---|---|---|---|
| Contract deployment | 674062 | 88 | 0.0559471 |
| Deposit() | 44020 | 88 | 0.0036537 |
| WithdrawEV() | 30383 | 88 | 0.0025218 |
| WithdrawCS() | 48162 | 88 | 0.0039974 |
| Payment() | 52727 | 88 | 0.0043763 |
| UpdateT() | 34120 | 88 | 0.002832 |

## 5.8 Conclusions

The untrusted centralized nature of energy markets and EV charging infrastructures results in several privacy and security threats to EV users' private information. In this work, we proposed a blockchain-based end-to-end privacy-preserving CS reservation and payment protocol, which enables EVs to reserve a charging slot at the selected CS privately, without sharing their private information or exposing their identity/blockchain address at CS premises, thus ensuring physical identity unlinkability on the blockchain network. Additionally, we provided a decentralized CS information verification protocol to mitigate possible charging slot conflict. Finally, we proposed a dispute resolution protocol among EVs and CSs to ensure fairness and trust in the EV charging network, and to enforce normal behavior in an untrusted environment. Security analysis demonstrates that the proposed protocol ensures the end-to-end privacy of EV owners with security against multiple attacks.

# CONCLUSION AND RECOMMENDATIONS

Finally, this section provides a summary of the findings of this thesis in Section 6.1, as well as an outlook on current and future work in Section 6.2.

## 6.1 Summary

Secure service provider selection methods for IoT applications did not get much attention in the past. This was sufficient for earlier IoT applications as there were a few service providers. However, this thesis argues that each individual IoT application is different when it comes to functional and non-functional requirements. At the same time many service providers are entering the market with differentiated business models and service abstractions, therefore, depending on the QoS requirements for IoT applications, one service provider's solution may be better than the other. Thus, a service provider should be selected, based on the specific needs and preferences of the end user IoT application. This thesis revolves around this hypothesis and makes concrete contributions in three different aspects.

In Chapter 3, IoT large-scale data storage use-case is presented, which needs to be stored and possess different service requirements in terms of data size, data retrieval frequency, price and performance requirements, and are specific to the individual IoT application. Therefore, a middleware is called upon, which performs the storage provider selection for the IoT application, based on the application's requirements and needs. These service requirements are captured in the form of a cost function, and consequently an optimization problem is formulated, which minimizes the overall cost associated with a service provider. To ensure the security, we integrate blockchain technology, which enforces the normal operability of the middleware, and allows IoT applications to verify the decision made by the middleware. Furthermore, to improve the computational complexity arises by the middleware design, we propose a NN-based maintenance strategy, which intelligently decides when to take the data storage decision, i.e., how often

we find the solution to the optimization problem. We evaluate our middleware design, and our results show that our middleware effectively reduces the price of IoT data storage, while maintaining service requirements. Moreover, the solution to the optimization problem provides relatively better accuracy compared to state-of-the-art with less computation time.

In Chapter 4, we investigate the problem of secure CS selection by EVs without sharing the private information to a central entity or CS. Traditionally, these decisions are taken by the central management system, which takes the EVs private information and schedule a charging station slot at selected CS. However, this poses serious security threats to EV privacy, and to solve this problem, we propose a decentralized CS selection mechanism. The service requirements of EVs are captured in the form of a cost function, and consequently an optimization problem is formulated, which minimizes the overall cost associated with a CS. We propose the use blockchain technology to ensure the partial anonymity of EV owner at the time of reserving a charging slot. EV owners reserves a charging slot remotely and pay for the availed services through a smart contract functionality on the blockchain network. We evaluate our proposed decentralized EV charging protocol with real-world New York CS dataset, and our results show that our middleware effectively reduces the charging service price for an EV, while minimizing the waiting time, charging time and travelling time. Moreover, as the user is not sharing any private information with any entity in the network, the proposed protocol is providing anonymity to the EV owner. Thus, compared to state-of-the-art schemes, our solution selects the best charging station by fulfilling the EV user's needs, while securing its identity.

Finally, in Chapter 5, we extend the work Chapter 4 and provides an end-to-end privacy-preserving mechanism to secure the identity of the EV owner in a long run. In Chapter 4, when an EV reaches CS to avail the charging service, CS can link the physical identity of the EV owner with the blockchain address at the time of payment. This can pose serious threats to EV owner security and privacy, as charging history can be retrieved with the blockchain address.

To solve this problem, we propose a blockchain-based private CS reservation protocol that allows EV owners to reserve a charging slot privately, without sharing their personal information or exposing their blockchain addresses at the CS. EVs collaborate using ring signatures to anonymously make a reservation at a selected CS. To enforce charging reservations between EVs and CSs, blockchain smart contracts are used. Additionally, a secure smart contract based payment mechanism is implemented using a time-lock protocol, which aggregates the payer's transactions into an obfuscated public ledger, such that no outside observer can identify the exact source of a blockchain transaction. We evaluate our proposed protocol, and our results show that the proposed protocol ensures end-to-end EV owners' privacy with low blockchain transaction and computation overhead.

## 6.2 Future work

We highlight some of the future work in both research directions.

### 6.2.1 Extending large-scale iot data storage selection

In our current middleware design for large-scale IoT data storage selection, both IoT application and storage technologies are storing their parameters on the blockchain network. The current design is not tailored to provide security to these parameters on the blockchain network. One direction can be considered as how to store these parameters on blockchain network in such a way that one can still verify the decision correctness of the middleware design. Moreover, currently middleware is solving an optimization problem for different IoT applications as the devices are power constraints and cannot perform huge calculations. Another direction can be to shift this decision capability to IoT devices itself, so they do not need to share the private data to storage technologies or the blockchain network. This can be done by a decentralized optimization problem or by proposing lightweight algorithms to find the solution. Finally, the parameters uploaded by the storage technologies are trusted values and cannot be verified. One

direction can be to propose a mechanism which allows IoT applications to audit and verify the validity of the parameters associated with the storage technologies. Once these parameters are validated, one can implement smart contract functionality to automatically impose penalty, if the storage technologies fail to meet the application's promised QoS.

### 6.2.2 Extending decentralized EV charging network

In Chapter 4, we propose a decentralized CS selection mechanism, which uses a pub/sub mechanism to push charging information to the EVs. This work assumes that the channel is reliable when CSs broadcast the CS information to RSUs and RSUs then forwards this information to EVs. However, this is not a straight forward method as it needs proper modelling on physical and MAC layer. This work can be extended by proposing a reliable channel for message broadcasting service. Moreover, the work also considers the reputation mechanism and stores reputation directly on the blockchain network. The work does not solve the problem if a malicious user provides wrong information about a certain EV and wrong reputation value is stored and cannot be deleted. This problem can be addressed in future, where a mechanism is provided to make sure that only correct reputation values are stored on the blockchain network. One can also stores these values off-chain on decentralized storage technologies, e.g., IPFS or Storj. Furthermore, the RSU has been considered trusted in this work. One can extend this work by considering RSU as a malicious entity and provide the same security guarantee to the EV owners.

In Chapter 5, we propose a decentralized CS reservation and payment mechanism, which uses a ring signature mechanism to reserve a charging slot and time-lock mixing smart contract to ensure anonymous payments. The protocol works well with large number of EV participants, however, in case of small number of EVs coming to the CS on a certain day or a period of days, one can see the balances of the addresses in the smart contract transaction history, which got changed over this period of time. A statistical correlation analysis of physical identities/plate

number with the blockchain address can compromise the EV owner's privacy, One research direction can be to solve this problem in future work. Moreover, the experiments in this work are performed on laptop however, in real-world scenarios, the EV owners will use their mobile phones or in-vehicle infotainment to make the charging reservations. The computation capability differs among these devices, therefore, the computation time can be increased or decreased depends on the device specifications in terms of performance. So, another research direction is to test the protocol on hardware to see the possible performance and security aspects of the proposed protocol. Also, in this work, using the large number of public keys to generate ring signature give rise to a performance limitation, as it may take more than a minute to reserve a charging slot. It is possible that two EVs want to reserve the same time slot at the same time, which could result in possible conflict. This problem can be resolved by proposing a solution in future work.

## 6.3 Other possible applications

The privacy-preserving selection services can also be proposed for other application including ride-sharing application and P2P prosumer energy markets.

### 6.3.1 Ride-sharing application

Ride sharing services assist drivers to find proper riders for vacant seats on the road, providing appealing benefits of shared travel cost and improved vehicle occupancy, which have revolutionized transportation business, as witnessed by the success of Lyft Line, UberPool, and Waze Carpool. To initialize a sharing trip, a driver sends a ride offer or a rider sends a ride request to a trusted ride sharing server for finding rider-share partners with similar itineraries. Selecting proper ride-share partners for drivers based on riders' trip data is essential for ride sharing services, but it also leads to the exposure of drivers' and riders' future locations and trajectories, as the trusted server may not be fully trusted, i.e., it has numerous motivations

to share the detailed trip data with their cooperators for monetary reasons. Therefore, it is necessary to select the driver based on the service requirements of the rider, while ensuring the location and identity privacy of the rider, with a minimal trust assumptions or without the use of trusted third-party servers. Moreover, a privacy-preserving payment mechanism among drivers and riders can also be considered to ensure anonymous riding history. Our proposed privacy-preserving selection mechanism can be used to provide solution to the above mentioned problems with proper modifications according to the application requirements.

### 6.3.2 P2P energy trading market

Recent years distribution systems have witnessed a rapid interconnection of distributed energy resources (DER), solar energy generation and EVs in particular. Electricity customers equipped with DERs are transforming their roles from pure electricity consumers to prosumers (they can also sell energy), thus enabling P2P energy markets. However, the decision making for the buyers to select an appropriate energy seller is not an easy task as many important fac- tors need to be considered including price, seller availability, seller-buyer distance, the credit score, and seller downtime etc. Moreover, the participating parties in the energy market cannot be trusted for buyer's and seller's private information, as this information can be used to track the vehicle's location or user's daily behavior. Therefore, it is necessary to select the seller based on the service requirements of the buyer, while ensuring the location and identity privacy of both parties. Our proposed privacy-preserving selection mechanism can be used to select the appropriate seller in a P2P energy market.

## AUTHOR'S PUBLICATIONS

During the course of his PhD research, the author contributed to the following published and submitted research articles.

[1] Syed Muhammad Danish, Kaiwen Zhang, and Hans-Arno Jacobsen. "BlockAM: An adaptive middleware for intelligent data storage selection for internet of things.", IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), (2020).

[2] Syed Muhammad Danish, Kaiwen Zhang, and Hans-Arno Jacobsen. "BlockAIM: a neural network-based intelligent middleware for large-scale IoT data placement decisions." IEEE Transactions on Mobile Computing (2021).

[3] Syed Muhammad Danish, Kaiwen Zhang, Hans-Arno Jacobsen, Nouman Ashraf, and Hassaan Khaliq Qureshi. "BlockEV: Efficient and secure charging station selection for electric vehicles." IEEE Transactions on Intelligent Transportation Systems (2020).

[4] Syed Muhammad Danish, Munim Shabir, Kaiwen Zhang, Hans-Arno Jacobsen, and Syed Ali Hassan. "A Blockchain-Based Privacy-Preserving Charging Station Reservation and Payment Scheme for Electric Vehicles", submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems (2022).

## BIBLIOGRAPHY


Aazam, M. & Huh, E.-N. (2016). Fog computing: The cloud-iot\/ioe middleware paradigm. *IEEE Potentials*, 35(3), 40–44.

Abbate, T., Cesaroni, F., Cinici, M. C. & Villari, M. (2019). Business models for developing smart cities. A fuzzy set qualitative comparative analysis of an IoT platform. *Technological Forecasting and Social Change*, 142, 183–193.

Abosaif, A. N. & Hamza, H. S. (2020). Quality of service-aware service selection algorithms for the internet of things environment: A review paper. *Array*, 8, 100041.

Aggarwal, S. & Kumar, N. (2021). Hyperledger. In *Advances in Computers* (vol. 121, pp. 323–343). Elsevier.

Al-Bassam, M., Sonnino, A., Król, M. & Psaras, I. (2018). Airtnt: Fair exchange payment for outsourced secure enclave computations. *arXiv preprint arXiv:1805.06411*.

Al-Hawawreh, M., Moustafa, N., Garg, S. & Hossain, M. S. (2020). Deep Learning-enabled Threat Intelligence Scheme in the Internet of Things Networks. *IEEE Transactions on Network Science and Engineering*, 8(4), 2968–2981.

Al-Rubaye, S., Al-Dulaimi, A. & Ni, Q. (2019a). Power interchange analysis for reliable vehicle-to-grid connectivity. *IEEE Communications Magazine*, 57(8), 105–111.

Al-Rubaye, S., Rodriguez, J., Al-Dulaimi, A., Mumtaz, S. & Rodrigues, J. J. (2019b). Enabling digital grid for industrial revolution: self-healing cyber resilient platform. *IEEE Network*, 33(5), 219–225.

AlZain, M. A. & et al. (2011). Mcdb: using multi-clouds to ensure security in cloud computing. *In DASC'11*, pp. 784–791.

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P. & Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100, 143–174.

Antoun, J., Kabir, M. E., Moussa, B., Atallah, R. & Assi, C. (2020). A Detailed Security Assessment of the EV Charging Ecosystem. *IEEE Network*, 34(3), 200–207.

Au, M. H., Liu, J. K., Fang, J., Jiang, Z. L., Susilo, W. & Zhou, J. (2013). A new payment system for enhancing location privacy of electric vehicles. *IEEE transactions on vehicular technology*, 63(1), 3–18.

B, I. & et al. (2008). Approximation algorithms for data placement problems. *SIAM Journal on Computing*, 38(4), 1411–1429.

Bai, J. & Hao, R. (2019). Comment on "Privacy-preserving public auditing for non-manager group shared data". *The Journal of Supercomputing*, 100(4), 1277–1294.

Bao, K., Valev, H., Wagner, M. & Schmeck, H. (2018). A threat analysis of the vehicle-to-grid charging protocol ISO 15118. *Computer Science-Research and Development*, 33(1-2), 3–12.

Barnaghi, P., Tönjes, R., Höller, J., Hauswirth, M., Sheth, A. & Anantharam, P. (2014). Citypulse: Real-time iot stream processing and large-scale data analytics for smart city applications. *Europen semantic web conference (ESWC)*, 2014.

Bayram, I. S. & Papapanagiotou, I. (2014). A survey on communication technologies and requirements for internet of electric vehicles. *EURASIP Journal on Wireless Communications and Networking*, 2014(1), 1–18.

Baza, M. & et al. (2021). Privacy-preserving blockchain-based energy trading schemes for electric vehicles. *IEEE Transactions on Vehicular Technology*, 70(9), 9369–9384.

Benisi, N. Z., Aminian, M. & Javadi, B. (2020). Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 102656.

Blum, J. J. & Eskandarian, A. (2007). A reliable link-layer protocol for robust and scalable intervehicle communications. *IEEE Transactions on Intelligent Transportation Systems*, 8(1), 4–13.

Bodet, C., Schülke, A., Erickson, K. & Jabłonowski, R. (2012). Optimization of charging infrastructure usage under varying traffic and capacity conditions. *IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 424–429.

Bowers, K. D. & et al. (2009). HAIL: A high-availability and integrity layer for cloud storage. *Proc. of the 16th ACM conference on Computer and communications security*, pp. 187–198.

Bryden, T. S., Hilton, G., Dimitrov, B., de León, C. P. & Cruden, A. (2019). Rating a stationary energy storage system within a fast electric vehicle charging station considering user waiting times. *IEEE Transactions on Transportation Electrification*, 5(4), 879–889.

Cachin, C., Haas, R. & Vukolic, M. (2010). Dependable storage in the intercloud. *IBM research*, 3783, 1–6.

Cao, Y., Tang, S., Li, C., Zhang, P., Tan, Y., Zhang, Z. & Li, J. (2011). An optimized EV charging model considering TOU price and SOC curve. *IEEE Transactions on Smart Grid*, 3(1), 388–393.

Cao, Y., Wang, N., Kamel, G. & Kim, Y.-J. (2015). An electric vehicle charging management scheme based on publish/subscribe communication framework. *IEEE Systems Journal*, 11(3).

Cao, Y., Jiang, T., Kaiwartya, O., Sun, H., Zhou, H. & Wang, R. (2019). Toward pre-empted EV charging recommendation through V2V-based reservation system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(5), 3026–3039.

Castro, M., Liskov, B. et al. (1999). Practical byzantine fault tolerance. *OsDI*, 99(1999), 173–186.

Celik, Z. B., Babun, L., Sikder, A. K., Aksu, H., Tan, G., McDaniel, P. & Uluagac, A. S. (2018). Sensitive Information Tracking in Commodity {IoT}. *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1687–1704.

Chauhan, S. S., Pilli, E. S. & Joshi, R. C. (2021). BSS: a brokering model for service selection using integrated weighting approach in cloud environment. *Journal of Cloud Computing*, 10(1), 1–14.

Chen, T., Zhang, B., Pourbabak, H., Kavousi-Fard, A. & Su, W. (2016). Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems. *IEEE Transactions on Smart Grid*, 9(4), 3563–3572.

Chen, W. & et al. (2019). Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things. *IEEE Internet of Things Journal*, 6(5), 8433–8446.

Christidis, K. & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, 4, 2292–2303.

Clarke, R. & Wigan, M. (2011). You are where you've been: the privacy implications of location and tracking technologies. *Journal of Location Based Services*, 5(3-4), 138–155.

Dai, H.-N. & et al. (2019). Blockchain for Internet of Things: A survey. *IEEE Internet of Things Journal*, 6(5), 8076–8094.

Danish, S. M. (2019). A blockchain-based adaptive middleware for large scale internet of things data storage selection. *Proc. of Middleware'19 Doctoral Symposium*, pp. 17–19.

Danish, S. M. & et al. (2020). BlockAM: An Adaptive Middleware for Intelligent Data Storage Selection for Internet of Things. *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pp. 61–71.

Danish, S. M., Lestas, M., Qureshi, H. K., Zhang, K., Asif, W. & Rajarajan, M. (2020a). Securing the LoRaWAN join procedure using blockchains. *Cluster Computing*, 23(3), 2123–2138.

Danish, S. M., Zhang, K. & Jacobsen, H.-A. (2020b). A Blockchain-Based Privacy-Preserving Intelligent Charging Station Selection for Electric Vehicles. *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–3.

Danish, S. M., Zhang, K., Jacobsen, H.-A., Ashraf, N. & Qureshi, H. K. (2020c). BlockEV: Efficient and Secure Charging Station Selection for Electric Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(7), 4194–4211.

Danish, S. M., Zhang, K. & Jacobsen, H.-A. (2021). BlockAIM: a neural network-based intelligent middleware for large-scale IoT data placement decisions. *IEEE Transactions on Mobile Computing*.

Debe, M. & et al. (2019). IoT public fog nodes reputation system: A decentralized solution using Ethereum blockchain. *IEEE Access*, 7, 178082–178093.

del Razo, V. & , H.-A. J. (2016). Smart charging schedules for highway travel with electric vehicles. *IEEE Transactions on Transportation Electrification*, 2(2), 160–173.

Dobre, D. & et al. (2014). Hybris: Robust hybrid cloud storage. *Proc. of ACM Symposium on Cloud Computing*, pp. 1–14.

Dorri, A. & et al. (2017). Towards an optimized blockchain for IoT. *Proc. of the second IoTDI'17*, pp. 173–178.

Eiza, M. H., Shi, Q., Marnerides, A. K., Owens, T. & Ni, Q. (2018). Efficient, Secure, and Privacy-Preserving PMIPv6 Protocol for V2G Networks. *IEEE Transactions on Vehicular Technology*, 68(1), 19–33.

Eskandarian, A., Chaoxian, W. & Chuanyang, S. (2019). Research Advances and Challenges of Autonomous and Connected Ground Vehicles. *IEEE Trans. on Intelligent Transportation Systems*, 22(2), 683–711.

Fraiji, Y., Azzouz, L. B., Trojet, W. & Saidane, L. A. (2018). Cyber security issues of Internet of electric vehicles. *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6.

G, B. & et al. (2019). Winning at the starting line: Joint network selection and service placement for mobile edge computing. *IEEE INFOCOM'19*, pp. 1459–1467.

Gao, F., Zhu, L., Shen, M., Sharif, K., Wan, Z. & Ren, K. (2018). A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks. *IEEE Network*, 32(6), 184–192.

Goli, M. & Eskandarian, A. (2014). A systematic multi-vehicle platooning and platoon merging: Strategy, control, and trajectory generation. *ASME 2014 Dynamic Systems and Control Conference*, 46193, V002T25A006.

Goyal, P., Sharma, A., Vyas, S. & Kumar, R. (2016). Customer and aggregator balanced dynamic Electric Vehicle charge scheduling in a smart grid framework. *IEEE ICEPES*, pp. 276–283.

Guha, S. & Khuller, S. (1999). Greedy strikes back: Improved facility location algorithms. *Journal of algorithms*, 31(1), 228–248.

Gusrialdi, A., Qu, Z. & Simaan, M. A. (2017). Distributed scheduling and cooperative control for charging of electric vehicles at highway service stations. *IEEE Transactions on Intelligent Transportation Systems*, 18(10), 2713–2727.

Hassan, M. U., Rehmani, M. H. & Chen, J. (2021). VPT: Privacy Preserving Energy Trading and Block Mining Mechanism for Blockchain based Virtual Power Plants. *arXiv preprint arXiv:2102.01480*.

HC, T. & et al. (2009). *Introduction to Algorithms, Third Edition* (ed. 3rd). The MIT Press.

He, T., Bai, Y. & Zhu, J. (2016). Optimal charging strategy of electric vehicles customers in a smart electrical car park.

He, T., Zhu, J., Zhang, J. & Zheng, L. (2018). An optimal charging/discharging strategy for smart electrical car parks. *Chinese Journal of Electrical Engineering*, 4(2), 28–35.

Imai, S. & et al. (2018). A Performance Study of Geo-Distributed IoT Data Aggregation for Fog Computing. *2018 IEEE/ACM UCC Companion*, pp. 278–283.

Jeong, S., Dao, N.-N., Lee, Y., Lee, C. & Cho, S. (2018). Blockchain Based Billing System for Electric Vehicle and Charging Station. *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 308–310.

Jiang, S., Zhang, X., Li, J., Yue, H. & Zhou, Y. (2020). Secure and privacy-preserving energy trading scheme based on blockchain. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6.

Jiang, T., Fang, H. & Wang, H. (2018). Blockchain-based Internet of vehicles: distributed network architecture and performance analysis. *IEEE Internet of Things Journal*, 6(3), 4640–4649.

Jin, C., Tang, J. & Ghosh, P. (2013). Optimizing electric vehicle charging: A customer's perspective. *IEEE Transactions on Vehicular Technology*, 62(7), 2919–2927.

Jin, R., Zhang, X., Wang, Z., Sun, W., Yang, X. & Shi, Z. (2019). Blockchain-Enabled Charging Right Trading Among EV Charging Stations. *Energies*, 12(20), 3922.

Junqueira, F. P., Reed, B. C. & Serafini, M. (2011). Zab: High-performance broadcast for primary-backup systems. *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, pp. 245–256.

Kamuni, V., Asfia, U., Sutavani, S., Sheikh, A. & Patel, D. (2019). Secure Energy Market against Cyber Attacks using Blockchain. *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1792–1797.

Karlsson, K. & et al. (2018). Vegvisir: A partition-tolerant blockchain for the internet-of-things. *IEEE ICDCS'18*, pp. 1150–1158.

Khaled, A. E., Helal, A., Lindquist, W. & Lee, C. (2018). IoT-DDL–device description language for the "T" in IoT. *IEEE Access*, 6, 24048–24063.

Khodari, M., Rawat, A., Asplund, M. & Gurtov, A. (2019). Decentralized firmware attestation for in-vehicle networks. *5th on Cyber-Physical System Security Workshop*, pp. 47–56.

Kim, M., Park, K., Yu, S., Lee, J., Park, Y., Lee, S.-W. & Chung, B. A secure charging system for electric vehicles based on blockchain. *Sensors*, 19(13), 3028.

Kirpes, B. & Becker, C. (2018). Processing electric vehicle charging transactions in a blockchain-based information system.

Knirsch, F., Unterweger, A. & Engel, D. (2018). Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. *Computer Science-Research and Development*, 33(1), 71–79.

Ko, R. K., Lee, B. S. & Pearson, S. (2011). Towards achieving accountability, auditability and trust in cloud computing. *International conference on advances in computing and communications*, pp. 432–444.

Kosba, A., Miller, A., Shi, E., Wen, Z. & Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *2016 IEEE symposium on security and privacy (SP)*, pp. 839–858.

Kumar, P. A. R. & Selvakumar, S. (2011). Distributed denial of service attack detection using an ensemble of neural classifier. *Computer Communications*, 34(11), 1328–1341.

Kumar, R. S. & Saxena, A. (2011). Data integrity proofs in cloud storage. *IEEE Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pp. 1–4.

L, N. & et al. (2015). Safecoin: The decentralised network token. MaidSafe.

Lambert, N. & Bollen, B. (2014). The SAFE Network: a New, Decentralised Internet.

Larson, D. (2016). Distributed denial of service attacks–holding back the flood. *Network Security*, 2016(3), 5–7.

Le, D. N., Le Tuan, L. & Tuan, M. N. D. (2019). Smart-building management system: An Internet-of-Things (IoT) application business model in Vietnam. *Technological Forecasting and Social Change*, 141, 22–35.

Lee, I. & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business horizons*, 58(4), 431–440.

Li, J., Zhou, Z., Wu, J., Li, J., Mumtaz, S., Lin, X., Gacanin, H. & Alotaibi, S. (2019). Decentralized on-demand energy supply for blockchain in internet of things: A microgrids approach. *IEEE Transactions on Computational Social Systems*, 6(6), 1395–1406.

Li, R., Song, T., Mei, B., Li, H., Cheng, X. & Sun, L. (2018). Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 12(5), 762–771.

Liu, W.-L., Gong, Y.-J., Chen, W.-N., Liu, Z., Wang, H. & Zhang, J. (2019). Coordinated Charging Scheduling of Electric Vehicles: A Mixed-Variable Differential Evolution Approach. *IEEE Transactions on Intelligent Transportation Systems*, 21(12), 5094–5109.

Long, Y., Chen, Y., Ren, W., Dou, H. & Xiong, N. N. (2020). Depet: a decentralized privacy-preserving energy trading scheme for vehicular energy network via blockchain and k-anonymity. *IEEE Access*, 8, 192587–192596.

Lu, X., Guan, Z., Zhou, X., Wu, L., Du, X. & Guizani, M. (2019). An efficient and privacy-preserving energy trading scheme based on blockchain. *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6.

Masood, A. B. & et al. (2019). Closing the Loop in Cyber-Physical Systems using Blockchain: Microgrid Frequency Control Example. *2019 2nd IEEE MENACOMM*, pp. 1–6.

Masood, A. B., Qureshi, H. K., Danish, S. M. & Lestas, M. (2019). Realizing an implementation platform for closed loop cyber-physical systems using blockchain. *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5.

Moghaddam, M. & Davis, J. G. (2014). Service selection in web service composition: A comparative review of existing approaches. *Web services foundations*, 321–346.

Mollah, M. & et al. (2017). Secure data sharing and searching at the edge of cloud-assisted internet of things. *IEEE Cloud Computing*, 34–42.

Mukherjee, J. C. & Gupta, A. (2014). A mobility aware scheduler for low cost charging of electric vehicles in smart grid. *2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–8.

Nakamoto, S. (2008). Re: Bitcoin P2P e-cash paper. *The Cryptography Mailing List*.

Nejad, M. M., Mashayekhy, L., Chinnam, R. B. & Grosu, D. (2017). Online scheduling and pricing for electric vehicle charging. *IISE Transactions*, 49(2), 178–193.

Nimalsiri, N. I., Mediwaththe, C. P., Ratnam, E. L., Shaw, M., Smith, D. B. & Halgamuge, S. K. (2019). A survey of algorithms for distributed charging control of electric vehicles in smart grid. *IEEE Transactions on Intelligent Transportation Systems*, 21(11), 4497–4515.

Nord, J. H., Koohang, A. & Paliszkiewicz, J. (2019). The Internet of Things: Review and theoretical framework. *Expert Systems with Applications*, 133, 97–108.

Ongaro, D. & Ousterhout, J. (2014). In search of an understandable consensus algorithm. *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, pp. 305–319.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. *International conference on the theory and applications of cryptographic techniques*, pp. 223–238.

Pajic, J., Rivera, J., Zhang, K. & Jacobsen, H.-A. (2018). Eva: Fair and auditable electric vehicle charging service using blockchain. *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*, pp. 262–265.

Papaioannou, T. G., Bonvin, N. & Aberer, K. (2012). Scalia: An adaptive scheme for efficient multi-cloud storage. *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–10.

Patil, H. & Kalkhambkar, V. N. (2019). Charging cost minimisation by centralised controlled charging of electric vehicles. *International Transactions on Electrical Energy Systems*, 30(2), e12226.

Pourazarm, S., Cassandras, C. G. & Malikopoulos, A. (2014). Optimal routing of electric vehicles in networks with charging nodes: A dynamic programming approach. *2014 IEEE International Electric Vehicle Conference (IEVC)*, pp. 1–7.

Psaras, Y. & Dias, D. (2020). The interplanetary file system and the filecoin network. *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pp. 80–80.

Pustišek, M., Kos, A. & Sedlar, U. (2016). Blockchain based autonomous selection of electric vehicle charging station. *2016 international conference on identification, information and knowledge in the Internet of Things (IIKI)*, pp. 217–222.

Pyrkova, A. & Temirbekova, Z. (2020). Compare encryption performance across devices to ensure the security of the IOT. *Indonesian Journal of Electrical Engineering and Computer Science*, 20(2), 894–902.

Qin, H. & Zhang, W. (2011). Charging scheduling with minimal waiting in a network of electric vehicles and charging stations. *Proceedings of the Eighth ACM international workshop on Vehicular inter-networking*, pp. 51–60.

R, J. & et al. (2019). Blockchain-Based Distributed Cloud Storage Digital Forensics: Where's the Beef? *IEEE Security & Privacy*, 17(1).

Radi, E. M., Lasla, N., Bakiras, S. & Mahmoud, M. (2019). Privacy-Preserving Electric Vehicle Charging for Peer-to-Peer Energy Trading Ecosystems. *IEEE International Conference on Communications (ICC)*, pp. 1–6.

Rafique, A. & et al. (2017). Towards an adaptive middleware for efficient multi-cloud data storage. *CrossCloud'17*, pp. 4.

Rafique, A. & et al. (2019). SCOPE: self-adaptive and policy-based data management middleware for federated clouds. *Journal of Internet Services and Applications*, 10(1), 2.

Rahman, M. S., Khalil, I., Alabdulatif, A. & Yi, X. (2019). Privacy preserving service selection using fully homomorphic encryption scheme on untrusted cloud service platform. *Knowledge-Based Systems*, 180, 104–115.

Ramnath, S., Javali, A., Narang, B., Mishra, P. & Routray, S. K. (2017). IoT based localization and tracking. *2017 International Conference on IoT and Application (ICIOT)*, pp. 1–4.

Rathee, G., Sharma, A., Iqbal, R., Aloqaily, M., Jaglan, N. & Kumar, R. (2019). A blockchain framework for securing connected and autonomous vehicles. *Sensors*, 19(14), 3165.

Rivera, J., Goebel, C. & Jacobsen, H.-A. (2016). Distributed convex optimization for electric vehicle aggregators. *IEEE Transactions on Smart Grid*, 8(4), 1852–1863.

Rivest, R. L., Shamir, A. & Tauman, Y. (2001). How to leak a secret. *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 552–565.

Rubio, J. E., Alcaraz, C. & Lopez, J. (2018). Addressing Security in OCPP: Protection Against Man-in-the-Middle Attacks. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5.

Ruiz-Alvarez, A. & Humphrey, M. (2011). An automated approach to cloud storage service selection. *Proceedings of the 2nd international workshop on Scientific cloud computing*, pp. 39–48.

Saghezchi, F. B., Mantas, G., Ribeiro, J., Al-Rawi, M., Mumtaz, S. & Rodriguez, J. (2017). Towards a secure network architecture for smart grids in 5G era. *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 121–126.

Said, D., Cherkaoui, S. & Khoukhi, L. (2013). Queuing model for EVs charging at public supply stations. *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 65–70.

Saied, A., Overill, R. E. & Radzik, T. (2016). Detection of known and unknown DDoS attacks using Artificial Neural Networks. *Neurocomputing*, 172, 385–393.

Samuel, O., Javaid, N., Shehzad, F., Iftikhar, M. S., Iftikhar, M. Z., Farooq, H. & Ramzan, M. (2019). Electric Vehicles Privacy Preserving Using Blockchain in Smart Community. *International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 67–80.

Samy, A., Yu, H., Zhang, H. & Zhang, G. (2021). SPETS: Secure and Privacy-Preserving Energy Trading System in Microgrid. *Sensors*, 21(23), 8121.

Schmidt, K., Saucke, F. & Spengler, T. S. (2018). Scheduling of Electric Vehicles in the Police Fleet. In *Operations Research Proceedings 2017* (pp. 693–699). Springer.

Shafagh, H. & et al. (2017). Towards blockchain-based auditable storage and sharing of iot data. *Proceedings of the 2017 on Cloud Computing Security Workshop*, pp. 45–50.

Shi, W. & Wong, V. W. (2011). Real-time vehicle-to-grid control algorithm under price uncertainty. *IEEE SmartGridComm*, pp. 261–266.

Son, Y.-B., Im, J.-H., Kwon, H.-Y., Jeon, S.-Y. & Lee, M.-K. (2020). Privacy-preserving peer-to-peer energy trading in blockchain-enabled smart grids using functional encryption. *Energies*, 13(6), 1321.

Squicciarini, A., Carminati, B. & Karumanchi, S. (2011). A privacy-preserving approach for web service selection and provisioning. *2011 IEEE International Conference on Web Services*, pp. 33–40.

Su, Z., Wang, Y., Xu, Q., Fei, M., Tian, Y.-C. & Zhang, N. (2018). A secure charging scheme for electric vehicles with smart communities in energy blockchain. *IEEE Internet of Things Journal*, 6(3), 4601–4613.

Sun, Y., Yin, L., Sun, Z., Tian, Z. & Du, X. (2020). An IoT data sharing privacy preserving scheme. *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 984–990.

Sweda, T. M. & Klabjan, D. (2012). Finding minimum-cost paths for electric vehicles. *IEEE International Electric Vehicle Conference*, pp. 1–4.

Syed, T. A., Alzahrani, A., Jan, S., Siddiqui, M. S., Nadeem, A. & Alghamdi, T. (2019). A comparative analysis of blockchain architecture and its applications: Problems and recommendations. *IEEE access*, 7, 176838–176869.

Tsaousoglou, G., Steriotis, K. & Varvarigos, E. (2019). A stochastic approximation method for price-based assignment of Electric Vehicles to Charging Stations. *IEEE International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1–6.

Tuan, M. N. D., Thanh, N. N. & Le Tuan, L. (2019). Applying a mindfulness-based reliability strategy to the Internet of Things in healthcare–A business model in the Vietnamese market. *Technological Forecasting and Social Change*, 140, 54–68.

Vaquero, L. M. & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5), 27–32.

Vorick, D. & Champine, L. (2014). Sia: Simple decentralized storage. *White paper available at https://sia. tech/sia. pdf.*

Wang, Q. & et al. (2009). Enabling public verifiability and data dynamics for storage security in cloud computing. *Proc. of ESORICS'09,*, pp. 355–370.

Wen, M., Linde, E., Ropke, S., Mirchandani, P. & Larsen, A. An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76, 73–83.

Wilkinson, S., Boshevski, T., Brandoff, J. & Buterin, V. (2014). Storj a peer-to-peer cloud storage network.

Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014), 1–32.

Wu, Z. & et al. (2013). Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. *Proc. of ACM Symposium on Operating Systems Principles*, pp. 292–308.

Xia, Y., Chen, P., Bao, L., Wang, M. & Yang, J. (2011). A QoS-aware web service selection algorithm based on clustering. *2011 IEEE International Conference on Web Services*, pp. 428–435.

Xu, S., Chen, X. & He, Y. (2021). EVchain: An Anonymous Blockchain-Based System for Charging-Connected Electric Vehicles. *Tsinghua Science and Technology*, 26(6), 845–856.

Yang, J. & Yang, Z. (2014). Pricing scheme for aggregate load scheduling of plug-in electric taxi fleet. *Proceedings of the 33rd Chinese Control Conference*, pp. 7589–7594.

Yang, S.-N., Cheng, W.-S., Hsu, Y.-C., Gan, C.-H. & Lin, Y.-B. (2013). Charge scheduling of electric vehicles in highways. *Mathematical and Computer Modelling*, 57(11-12), 2873–2882.

Yang, W., Guan, Z., Wu, L., Du, X., Lv, Z. & Guizani, M. (2020). Autonomous and Privacy-preserving Energy Trading Based on Redactable Blockchain in Smart Grid. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6.

Yi, H., Guan, K., He, D., Ai, B., Dou, J. & Kim, J. (2019). Characterization for the Vehicle-to-Infrastructure Channel in Urban and Highway Scenarios at the Terahertz Band. *IEEE Access*, 7, 166984–166996.

Yu, Y., Song, T., Su, C., Tang, X. & Han, Z. (2019). Hierarchical Game for Electric Vehicle Public Charging Market. *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6.

Z, J. & et al. (2019). Data Centers Selection for Moving Geo-distributed Big Data to Cloud. *Journal of Internet Technology*, 20(1), 111–122.

Zhang, K. & et al. (2018). Deconstructing Blockchains: Concepts, Systems, and Insights. *DEBS*, pp. 187–190.

Zhang, K. & Jacobsen, H.-A. (2018). Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains (Technical Report).

Zhang, Y., Xu, C., Lin, X. & Shen, X. S. (2019). Blockchain-based public integrity verification for cloud storage against procrastinating auditors. *IEEE Transactions on Cloud Computing*, 9(3), 923–937.

Zhou, Z., Xiong, F., Xu, C., He, Y. & Mumtaz, S. (2017). Energy-efficient vehicular heterogeneous networks for green cities. *IEEE Transactions on industrial Informatics*, 14(4), 1522–1531.