People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research
University Mohammed Seddik Ben Yahia University of Jijel

Faculty of Exact Sciences and Computer Science

Department of Computer Science

**Master Report**

for partial fulfillment of the requirements for the

Master's degree in computer science

**Option:** Network and Security

---

# A blockchain-based trust management system to secure task offloading in mobile edge computing.

---

**Prepared By :**
MATI Randa
BOUCHEMAL Nesrine

**Supervised by :**
Dr. ALIOUA Ahmed

*Academic year*
*2021-2022*

# *Dedication*

***To my dear parents, ,***

There isn't really a word I can use to express how grateful I am to have you as a parent in my life, and nothing I do or say will ever be enough to thank you. You've given me a lot of love and support throughout my life, and you're always there for me when I need it. You've always been at my side, encouraging and supporting me in every decision I've made in my life. You are my heroes and my example in life. May God, the Most High, provide you health, happiness, and a long life, and may I never deceive you. That this work expresses my thanks and affection. .

To my brother for your support and encouragement, as well as the joy you bring me every day with your humor, To my wonderful sisters for their moral support and encouragement throughout my life.

**To my binome** ,

I'd also want to thank my binome, who is also my best friend, for her support and for always making me happy, for never leaving me alone, and for being the one who follows me in all my decisions.

**To all my friends** .

I am grateful to have you as a friend, I love you.

*Mati Randa*

# *Dedication*

***With the expression of my gratitude, I dedicate this work To:***

To my parents, my precious gift from God, who owes my life and my success, who gave me a good education. Nothing can ever be enough to express my gratitude to you. May God bless you with good health and long life

To my dear sisters WARDA and MAROUA who have never stopped to encourage and support me throughout my studies. May God protect them and give them luck and happiness.

To my adorable little brother HICHEM who always knows how to bring joy and happiness to the whole family.

To my grandmother and my grandfather. May God give them a long and happy life.

To my binom and my best friend RANDA, for her support, her patience and for beign always there for me, as a true friend. I'm so lucky that I have you!

To all my friends who have always encouraged me and to whom I wish more success.

*Bouchemal Nesrine*

# *Acknowledgement*

First and foremost, we thank **Allah**, who has given us the courage, the will, the strength, the health and the persistence to accomplish this modest work. Thank you for showing us the way to success.

We thank our supervisor **Dr. ALIOUA Ahmed**, who has been very available throughout the realization, and for his help, guidance, and innumerable advices.

We also owe special thanks to our teachers and colleagues at University of Jijel. Our heartfelt thanks goes to the jury members for the honor of accepting to judge this work and for all of their insightful comments.

We are grateful to everyone who has contributed in any way to the completion of this work, our teachers, our parents, and our brothers, sisters, our friends.

## Abstract

With the growth of the Internet of Things (IoT), the majority of apps run on mobile devices are increasingly reliant on artificial intelligence (AI) algorithms, such as approaches for automatic learning, image/video processing, augmented reality, online gaming, and the long-awaited arrival of metaverse applications. These applications need a large amount of resources in order to perform extremely complicated calculations, mobile terminals face significant limits in terms of computation and storage capacity, in addition to a limited battery life. As a solution, computation / data offloading has been presented as a possible resource allocation strategy that comprises transferring intensive jobs and massive volumes of data to be performed by the mobile edge computing (MEC).

However, the lack of confidence between mobile terminals and MEC servers is seen as a serious security and confidentiality problem to the computation/data offloading approach's deployment. Indeed, before engaging in any interaction, each entity of the system must verify and confirm the reliability and the security of the other entity. Blockchain has recently been identified promising solution for increasing the security of MEC systems. Furthermore, the blockchain can provide high levels of security and trust for MEC by depending on a vast community's distributed verification of the authenticity of MEC system communicators via highly trustworthy consensus protocols.

Motivated by the previous discussion, we investigate in this work a trust management system to ensure more secure and reliable interactions in the MEC domain for task offloading. Our system is built on blockchain technology and consists primarily of two parts: a bidirectional trust management mechanism based on a reputation metric and an incentive approach that uses game theory to ensure more efficient block mining. We used a Stackelberg-type game to model the incentive problem in verification with asymmetric information in which the players are: a leader who acts first, followed by several followers. Our Stackelberg game is being played between edge devices.

The leader begins by announcing its strategy, after which the followers compete and respond with their best strategy that maximizes the leader's approach. After receiving all of the followers' replies, the leader optimizes its final strategy. In the last part we tried to show the relationship between the utility of the players (leader/follower) and the other parameters, and we obtained logical results.



**Keywords:** Mobile Edge Computing, Offloading, Blockchain, Incitation Mechanism, Game Theory

## Résumé

Avec la croissance de l'Internet des objets (IoT), la majorité des applications exécutées sur les terminaux mobiles dépendent de plus en plus des algorithmes d'intelligence artificielle (IA), comme les approches d'apprentissage automatique, de traitement d'images/vidéos, de réalité augmentée, de jeux en ligne, et l'arrivée tant attendue des applications metaverses. Ces applications nécessitent fréquemment une grande quantité de ressources afin d'effectuer des calculs extrêmement compliqués. Les terminaux mobiles sont confrontés à des limites importantes en termes de capacité de calcul et de stockage, en plus d'une autonomie limitée. Comme solution, le déchargement des calculs et des données (computation / data offloading) a été présenté comme une stratégie possible d'allocation des ressources qui consiste à transférer les calculs intensifs et les données intensives à effectuer à distance sur des serveurs distants de l'informatique mobile périphérique (mobile edge computing (MEC)).

Cependant, le manque de confiance entre les terminaux mobiles et les serveurs MEC est considéré comme un grave problème de sécurité et de confidentialité pour le déploiement de l'approche de déchargement des calculs/données. En effet, avant de s'engager dans une quelconque interaction, chaque composant du système doit vérifier et confirmer la fiabilité et la sécurité de l'autre composant. La blockchain a récemment été identifiée comme un concurrent possible pour accroître la sécurité des systèmes MEC. En outre, la blockchain peut fournir des niveaux élevés de sécurité et de confiance pour les MEC en dépendant de la vérification distribuée par une vaste communauté de l'authenticité des communicateurs du système MEC via des protocoles de consensus hautement fiables.

Nous concentrons nos efforts pour proposer un système de gestion de la confiance afin d'assurer des interactions plus sûres et plus fiables dans le domaine MEC pour le déchargement de calcul des tâches. Notre système est construit sur la technologie blockchain et se compose principalement de deux parties : un mécanisme de gestion de la confiance bidirectionnelle basé sur une métrique de réputation, une approche incitative qui utilise la théorie des jeux pour assurer une vérification plus efficace des blocs. Nous avons utilisé un jeu de type Stackelberg pour modéliser le problème d'incitation à la vérification avec information asymétrique dans lequel les joueurs sont : l'agent de calcul de tache (initiateur de block),le verificateurs de block.Un leader qui commence par annoncer sa stratégie, après les suiveurs (les vérificateurs de block) entrent en compétition et répondent avec leur meilleure stratégie qui maximise l'approche du leader. Après avoir reçu toutes les réponses des suiveurs, le leader optimise sa stratégie finale.

Dans la dernière partie, nous avons essayé à travers des différente expérimentations de montrer la relation entre l'utilité des joueurs (leader/suiveur) et les autres paramètres, et nous avons obtenu des résultats logiques.



**Mots clés :** Mobile Edge Computing, Offloading, Blockchain ,Théorie des jeux, Mechanisme d'incitation.

# Contents

**Abstract** i

**Résumé** ii

**List of Figures** viii

**List of Tables** x

**Acronyms** xi

**Introduction** 1

**1 An overview on computation offloading in mobile edge computing** 3

1.1 Introduction . . . 3

1.2 Cloud computing . . . 4

1.2.1 Definition . . . 4

1.2.2 Main characteristics . . . 4

1.2.3 Service models . . . 5

1.2.4 Deployment models . . . 6

1.2.5 Benefits and Challenges . . . 6

1.2.5.1 Benefits . . . 6

1.2.5.2 Challenges . . . 7
1.3 Mobile edge computing . . . 7
1.3.1 Definition . . . 8
1.3.2 Architecture . . . 8
1.3.3 Edge computing vs Cloud computing . . . 9
1.4 Towards computation offloading in mobile edge computing . . . 9
1.4.1 Computation offloading . . . 9
1.4.2 Types of computation offloading . . . 10
1.4.3 What to do before offloading ? . . . 10
1.4.3.1 Can the task be offloaded : what to offload? . . . 10
1.4.3.2 When to offload the task . . . 10
1.4.3.3 Where to offload . . . 11
1.4.3.4 Which offload policy apply . . . 11
1.4.3.5 What about trust (security) . . . 11
1.5 Conclusion . . . 11

**2 When trust management meets blockchain** **13**

2.1 Introduction . . . 13
2.2 Trust management . . . 14
2.2.1 Definition . . . 14
2.2.2 Composition . . . 14
2.2.3 Types . . . 15
2.2.4 Key concepts . . . 15
2.2.5 Properties of trust relationships . . . 15
2.2.6 Trust management in Mobile edge computing . . . 16
2.3 Blockchain technology . . . 16

2.3.1 History . . . . . 16
2.3.2 Definition . . . . . 17
2.3.3 Main characteristics . . . . . 18
2.3.4 Components . . . . . 19
2.3.5 Architecture . . . . . 20
2.3.6 Operating mode . . . . . 21
2.3.7 Types . . . . . 22
2.3.7.1 Public blockchain (permissionless blockchain) . . . . . 22
2.3.7.2 Private blockchain (permissioned blockchain) . . . . . 22
2.3.7.3 Consortium blockchain . . . . . 23
2.3.7.4 Hybrid blockchain . . . . . 23
2.3.8 Consensus models . . . . . 24
2.3.8.1 PoW (Proof of Work) . . . . . 24
2.3.8.2 PoS (Proof of Stake) . . . . . 24
2.3.8.3 DPoS (Delegated Proof of Stake) . . . . . 25
2.3.9 Challenges . . . . . 25
2.3.10 BC-based trust management . . . . . 26
2.4 Conclusion . . . . . 27

**3 BLOCKCHAIN-BASED TRUST MANAGEMENT SYSTEM FOR TASK OFFLOADING IN MEC 28**

3.1 Introduction . . . . . 28
3.2 Problem formulation . . . . . 29
3.2.1 Network Model . . . . . 29
3.2.2 System model . . . . . 31
3.3 A secure trust management protocol for task offloading in MEC . . . . . 32

3.3.0.1 Type of packets . . . 32
3.3.0.2 Operating Phases . . . 33
3.3.1 Construction of blockchain . . . 36
3.3.2 DPoS-based Consensus scheme . . . 37
3.4 BC-based incentive mechanism for block verification . . . 39
3.5 Stackelberg game-based incentive mechanism for block verification . . . 40
3.5.1 TCA's utility model . . . 41
3.5.1.1 Reward . . . 41
3.5.1.2 Cost . . . 42
3.5.2 Verifiers' utility model . . . 42
3.5.2.1 Reward . . . 43
3.5.2.2 Cost . . . 43
3.5.3 Follower's non-cooperative Sub-game . . . 44
3.6 Stackelberg equilibrium . . . 44
3.6.1 Follower's optimal strategy . . . 44
3.6.2 Leader optimal strategy . . . 49
3.6.3 Pseudo Algorithm to find the Stackelberg equilibrium . . . 51
3.7 Conclusion . . . 52

**4 Implementation and Simulation Results** **53**

4.1 Introduction . . . 53
4.2 Simulation tools and parameters . . . 53
4.2.1 Simulation language: Python . . . 53
4.2.2 Simulation tool: visual studio code (VS Code) . . . 54
4.2.3 Simulation hardware configuration . . . 54
4.3 Simulation parameters . . . 55

4.4 Presentation of the application interface . . . . . 55
4.5 Simulation results . . . . . 56
4.5.1 Impact of the number of verifiers on the utility function of the leader (TCA ) . . . . . 56
4.5.2 Impact of varying verifiers' reputations on the verifiers' utility . . . . . 57
4.5.3 Impact of varying verifiers' reputations on the TCA's utility . . . . . 58
4.5.4 Impact of the total incentive on the verifiers' utility . . . . . 58
4.5.5 Impact of the price ($\pi_j$) on the verifiers' utility . . . . . 59
4.5.6 Impact of the price ($\pi_j$) earned by the verifiers on TCA's utility . . . . . 60
4.5.7 Impact of the number of verifiers on the threshold and latency . . . . . 60
4.6 Conclusion . . . . . 61

**Conclusion** **62**

**A Game Theory** **63**

.1 Definition . . . . . 63
.2 Best response function . . . . . 64
.3 Sequential Game . . . . . 64
.3.1 Stackelberg Game . . . . . 64

**Bibliography** **66**

# List of Figures


1.1 Service models in cloud computing: IaaS, PaaS, SaaS. . . . . 5

1.2 Deployment models of Cloud computing. . . . . 6

1.3 An example of a basic three-layered MEC architecture [1]. . . . . 8

2.1 Simplified example of a Blockchain. . . . . 18

2.2 Layered blockchain Architecture.. . . . . 20

2.3 An illustrative scenario of how the Blockchain works [2]. . . . . 21

3.1 The network architecture of the BC-based computation offloading trust management system. . . . . 30

3.2 The architecture of the proposed BC-based trust management system. . . . . 32

3.3 The generic blockchain. . . . . 37

3.4 Two stages Stackelberg game for BC-based incentive caching . . . . . 41

4.1 Logo of Python. . . . . 54

4.2 Logo of Visual Studio Code. . . . . 54

4.3 The main application interface on python. . . . . 56

4.4 Impact of the number of verifiers on the utility of the leader (TCA). . . . . 57

4.5 Impact of reputation variation on the utility of verifiers. . . . . 57

4.6 Impact of reputation variation on the utility function of the leader ( TCA ). . . . 58

4.7 Impact of the total incentive on the utility function of verifiers. . . . . 59

4.8 Impact of the price on the utility function of verifiers. . . . . 59

4.9 Impact of the price on the utility function of TCA. . . . . 60

4.10 Impact of the varying number of verifiers on the threshold and latency. . . . . 61

# List of Tables

2.1 Taxonomic recapitulating summary of existing related works. . . . . 27

4.1 Configuration and setup of the used machine in simulation. . . . . 55

4.2 Simulation parameters. . . . . 55

# Acronyms

- **MEC :** Mobile Edge computing .
- **IoT :** Internet of Things .
- **GT :** Game Theory .
- **SG :** Stackelberg Game .
- **IT :** Information Technologies .
- **NE :** Nash equilibrium.
- **SE :** Stackelberg equilibrium.
- **KKT :** Karush-Kuhn-Tucker.

# Introduction

The Internet of Things (IoT) technology is exploding as we increasingly require intelligent objects in our daily lives to help us achieving our goals. It is currently seen as the next step in the growth of the Internet. IoT is no longer about linking tablets, laptops, and phones, but about allowing any aspect of the physical world to interact with one another, thereby eradicating the limits between physical objects and the virtual world. Unfortunately, exploiting the potential of IoT data is not simple. IoT devices are designed to stay connected while producing large volumes of data. As a result, collecting and analyzing sensor data from devices is a big challenge [2]. The concept of mobile edge computing (MEC) has emerge as a solution to the IoT device's limitations by offloading substantial computations and data to mobile cloud computing for processing and storage of huge amounts of data [3]. Simultaneously, with the emergence of the IoT and upcoming Artificial Intelligence (AI) applications, devices need to offload data to the cloud while being aware that the cloud is located in a geographically distant location. Hence, it will lead to a long response time for the user, especially since the timeout condition will endanger one's life in numerous scenarios such as self-driving cars and health related applications. For this purpose, MEC has arrived to eliminate these problems, which is a type of Network architecture that functions as an alternative to Cloud Computing. Rather than transmitting data treatment / computation offloading created by connected IoT devices to the cloud or a data center, the data treatment / computation offloading is processed at the network's edge where it is generated. However, trust issues exist at both the service provider and the mobile device levels, with mobile consumers concerned that mobile edge computing would fake their bills, miscalculate, or steal their data. On the MEC side, they don't know what data treatment / computation offloading the mobile device would offload, so it may be malicious data. Although certain trust management systems can ensure trust between two parties, they are systems of a third party, and there is no guarantee that they will not reveal private information. However, with the emergence of Blockchain technology, which is highly promising due to its anonymity, immutability, and decentralized nature, new trust management solutions based on this technology have simplified matters. Each system entity can interact with each other without the necessity of a third party.

**Objectives and contributions**

In this context, we are interested in presenting a new blockchain-based trust management system for securing the offloading process in mobile edge computing. Our system guarantees the confidentiality of the entities' identity in the system and also the security and reliability of the offloading process. The main contribution of this work is summarized as follow:

- We review the key works on blockchain based trust management for task offloading.
- We propose a blockchain-based trust management system, which improves the security of communications between mobile devices and mobile edge computing in untrusted environments by providing a higher level of privacy and reliability.
- We propose an incentive mechanism to encourage the different entities of the system to actively participate in our blockchain-based trust management system. For this, we propose a new strategy called tokens. Each user has a digital credit account to store his tokens. Initially, the system distributes a set of tokens to each entity on the system who has an account so that he can participate as verifier or auditor.
- We use a stackelberg game to model the strategy of interactions between a task computing agent and verifiers, the task computing agent proposes smart contracts to encourage verifiers with higher reputation scores to participate in the verification process, so that each verifier is incentivized to verify and validate the block in the shortest possible time. we use a concave optimization and backward induction to prove the stackelberg equilibrium and derive the optimal system strategies.
- We implement a series of extensive experimentation to evaluate the efficiency of our game theoretic-based verification incentive scheme.

**Organization of the report**

In addition to the introduction and conclusion, our report is structured in four chapters.

In chapter 1, we present an overview of computation offloading in MEC and Cloud computing.

In chapter 2, we present the fundamental concepts of the blockchain and the trust management. Then, we review the literature on blockchain-based trust management.

chapter 3 we present and proposed blockchain-based trust management system before detailing the game-theoretic-based incentive mechanism.

Finally, chapter4 describes the implementation and performance evaluation of our incentive system and the interpretation of the obtained results .

CHAPTER 1

# An overview on computation offloading in mobile edge computing

## 1.1 Introduction

The popular social media platforms, such as Facebook, Instagram, Twitter, YouTube, and Netflix, are currently Cloud-based. Furthermore, consumers are increasingly combining various mobile devices to satisfy their personal and professional needs. One of the primary benefits of the Cloud is the ability for users to access their data or computations at any time from any device, and it also enables devices with limited storage capacity to perform activities with high storage intensity by employing virtual storage. This not only reduces device prices but also improves battery life and overall user experience.

Simultaneously, Cloud computing is located in a geographically distant location. So it will lead to a long response time for the user. To resolve the problem of network latency, a new paradigm known as mobile edge computing (MEC) has been developed.Hence, The devices may communicate fastly with the nearest edge node, and may significantly reduce communication latency.

Nowdays mobile devices are ressource contrainted. As a solution to mobile devices' limited resources, offloading tasks to a distant server (Cloud computing or MEC) propose as a promising solution with various benefits, such as extending battery life, reducing latency, and enhancing application performance.

Throughout this chapter, we attempt to provide a comprehensive overview of the state of the art related to Cloud, mobile edge computing and computation offloading.

## 1.2 Cloud computing

The concept of Cloud computing first appeared in 1961, when professor John McCarthy postulated that shared computer technology may lead to a future in which processing power and even particular programs could be sold as a public utility. The word «Cloud computing» began to arise in technology circles during this time of change [3].

### 1.2.1 Definition

According to the official National Institute of Standards and Technology (NIST) definition « Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction» [3].

It is possible to define Cloud computing as the delivery of different services via the Internet such as processing power, storage, etc. Instead of buying, owning, and managing physical servers and data centers, you may use your own name to gain a remote access to Cloud-based technology services, and you pay based on the services used (pay-as-you-go).

### 1.2.2 Main characteristics

NIST has defined five key characteristics for Cloud [4]:

- **On-demand self-service :** A consumer can unilaterally provide computing capabilities such as server time and network storage, as required, without requiring human interaction with the provider of each service.
- **Broad network access :** The resources provided in the Cloud environment require access via the network through several types of devices (LapTops, PDAs, Tablets, Smart phones, etc.).
- **Resource pooling :** It signifies that the Cloud provider used a multi-tenant architecture to pull computing resources and to deliver storage services to different clients. The various physical and virtual resources are dynamically assigned and reassigned according to the requests of the customers. In general, customers have no control or knowledge of the actual location of the proposed resources; however, they can specify the location at a high level of abstraction such as country, state or datacenter.
- **Rapid elasticity :** A Cloud service must allow the user to deploy new resources or adjust service limitations almost instantly.

- **Measured service:** Cloud service providers are able to measure the consumption of different resources (CPU, Storage, Bandwidth, etc.) in a precise way. This measurement allows them to bill the customer according to the resources used.

### 1.2.3 Service models

There are three models of Cloud service [4]:

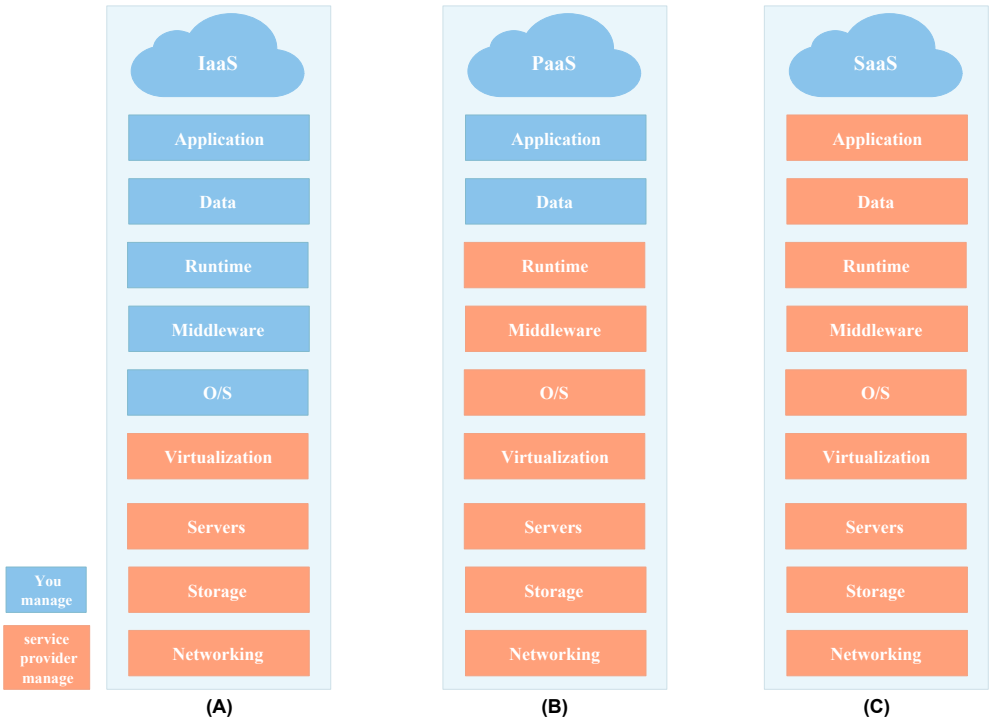


Figure 1.1: Service models in cloud computing: IaaS, PaaS, SaaS.

- **Infrastructure as a Service (IaaS) :** Provides users with access to basic computing resources such as processing capacity, data storage and networking, in a secure data center environment, see figure 1.1.(A).
- **Platform as a Service (Paas) :** The user is able to create and deploy their own applications using the provider's languages and tools. The user does not manage or control the underlying Cloud infrastructure (networks, servers, storage) but the user controls the deployed application and its configuration, see figure1.1.(B).
- **Software as a Service (SaaS) :** Offer application services adapted to the various needs of companies such as customer relationship management, marketing automation or value and profitability analysis, see figure 1.1.(C).

### 1.2.4 Deployment models

According to NIST there are four deployment models of Cloud computing, see 1.2 These deployment models are as follow [4]:

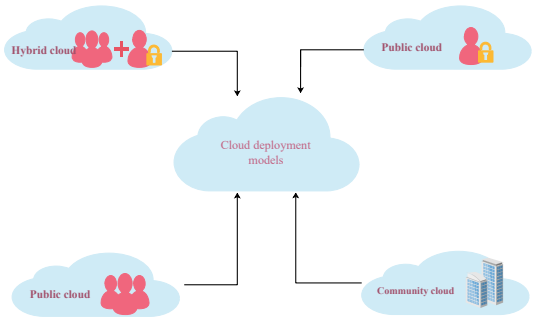


Figure 1.2: Deployment models of Cloud computing.

- **Public Cloud :** The public Cloud is a type of Cloud that provides various types of services over the Internet. These are typically provided by an organization that manages the infrastructure of the Cloud.
- **Private Cloud :** The infrastructure of the Cloud is used by a single organization. It can be managed by the organization or by a third party. The infrastructure can be placed on the organization's premises or externally.
- **Hybrid Cloud :** The Cloud infrastructure consists of two or more types of Clouds. These infrastructures are linked by the same technology, allowing for application and data portability.
- **Community Cloud :** Cloud infrastructure is shared by a group of organizations for the purposes of a community that wishes to share resources (security, compliance, etc.). It can be managed by the organizations or by a third party and can be located inside or outside.

### 1.2.5 Benefits and Challenges

Cloud computing, like any other technology, offers several advantages, whether technical, architectural or commercial. This does not mean that the Cloud has no issues and challenges.

#### 1.2.5.1 Benefits

Among the key benefits of Cloud computing, we cite the following:

- **Cost reduction :** Because no investment in physical hardware is required, significant capital cost savings can be achieved. It also reduces the need for skilled technicians to service equipment. A Cloud service provider owns and manages the equipment [5].

- **High speed :** Cloud computing allows services to be launched in less time than it takes to describe it. This faster deployment means that you can have the resources you need for your system in a very short time [6].

- **Quick deployment :** If you choose to use the Cloud, you can have your entire system up and running in minutes. However, the amount of time required depends on the type of technology used by the company [6].

- **Easy scalability :** Cloud computing is a flexible technique that allows for on-demand business scalability by using on-demand Cloud services such as SaaS, PaaS, or IaaS [5].

#### 1.2.5.2 Challenges

The main limits of Cloud computing are as follows:

- **Security and Privacy :** One of the biggest problems of Cloud computing is security. Cloud service providers host a large amount of data on their servers and many people and companies are concerned that data stored on remote sites might be hacked [5].

- **Reliability :** Because most organizations increasingly rely on third-party services, Cloud-based systems must be dependable and robust [5].

- **Interoperability :** This means that an application on one platform should be able to use services from other platforms. It is made feasible through online services, however implementing such web services is quite difficult [5].

- **Long delay :** The terminal is generally far from the Cloud server, and long-distance data transmission increases the transmission delay, which does not meet the requirements of real-time, low latency and high quality of service (QoS) in the network [7].

In recent years, a new computing technology known as Mobile edge computing (MEC), has emerged to avoid the transmission of large amounts of irrelevant data to data centers or the Cloud. In this second part we will present an overview of the MEC.

## 1.3 Mobile edge computing

The growth in the number of mobile devices and their associated applications that require extensive data exchange and powerfull processing has led to the development of Mobile-edge computing.

### 1.3.1 Definition

The term "Mobile edge computing", generally shortened as MEC, does not have a specific definition and there is no consensus in literature. Bellow we give two definition among the available definitions :

- According to the European Telecommunications Standards Institute (ETSI), MEC is defined as follows. “Mobile edge computing provides an IT service environment and Cloud computing capabilities at the edge of the mobile network, within the radio access network (RAN) and in close proximity to mobile subscribers.” [8].
- Another definition of MEC given by Arif and Ejaz in [9] “Mobile Edge computing is a model for enabling business oriented, Cloud computing platforms within the radio access network at the close proximity of mobile subscribers to serve delay sensitive, context aware applications.” [9].

In fact, MEC refers to a network architecture in which computing resources, storage capacity, and computing power are maintained as close as possible to the end devices.

### 1.3.2 Architecture

As illustrated in 1.3, mobile edge computing is composed of three distinct layers, which are explained below [1] :

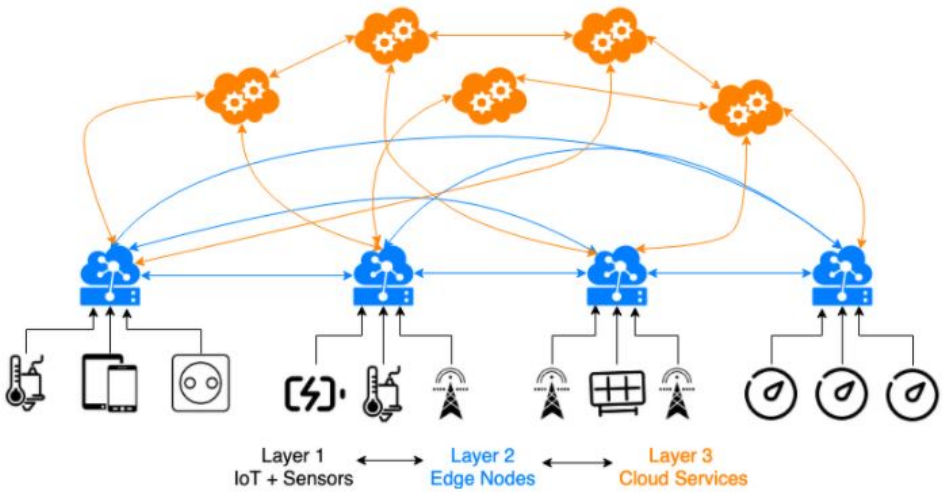


Figure 1.3: An example of a basic three-layered MEC architecture [1].

**Layer 1 - Edge devices :** This layer includes both IoT mobile devices (sensors, smart phones, smart plugs, smart vehicles, etc.) and users. The ingestion of data and the operations that go with it are handled by this first layer.

**Layer 2 - Edge nodes :** This layer is composed of edge nodes. These nodes handle data processing, routing, and computation.

**Layer 3 - Cloud services :** This layer consists of many Cloud services with higher computational requirements, it supports data analysis, artificial intelligence, machine learning and visualization.

### 1.3.3 Edge computing vs Cloud computing

While both edge and Cloud computing solutions are agile, scalable, dependable, and secure, they also boost productivity and performance, these two computing systems are fundamentally different. First, with Cloud computing, data is increasingly centralized in outsourced data centers, therefore edge solutions move their analytics and computing capabilities as near to the data source as possible, and also the edge computing minimizes service latency and network bandwidth compared with the Cloud [10].

Another difference is the cost. Cloud computing reduces it. It is not essential to invest in equipment or to assure the material's maintenance. Edge computing, on the other hand, is more expensive since it necessitates the installation and maintenance of equipment [11] [8]. Furthermore, Cloud computing often has a longer response time than edge computing.

## 1.4 Towards computation offloading in mobile edge computing

The majority of mobile applications are increasingly reliant on artificial intelligence (AI) algorithms, and these devices have limited battery and processing power, so when the edge node lacks processing capability, the computation task can be transferred to the edge server or the Cloud data center to improve overall system performance. This is known as "computation offloading" [12], [13].

### 1.4.1 Computation offloading

Computation offloading is a popular technique for improving mobile application performance while reducing power consumption by transferring computing tasks to external sources such as Clouds, edge, grids, or clusters, allowing the application to take advantage of powerful hardware and a sufficient power supply to increase responsiveness while reducing battery power consumption [14], [15].

### 1.4.2 Types of computation offloading

The concept of computation offloading appears to be very complicated and can be primarily divided into three types: full offloading, partial offloading, and local computation [12].

- **Full offloading :** The whole program, including all associated data, will be offloaded to an offloading server (Cloud or MEC server), where the computation completed, the results will be returned to the mobile device [12].
- **Partial offloading :** To improve the user experience, partial offloading refers to offloading a part of the user's tasks to be computed on an edge server and completing the remaining part locally. Usually, the part of the program that requires more energy or has greater computational complexity will be offloaded to the Cloud/edge server. Both the mobile device and the Cloud/edge are responsible for the calculation in this case, and the final results are obtained by combining the separate results of the computations performed on the mobile device and at the offloading server [16].
- **Local computation :** The entire calculation process is completed locally at the end users. This kind of computation is generally designed for tasks with low computing power requirements [17].

### 1.4.3 What to do before offloading ?

Before offloading the task computation, we need to answer a few questions :

#### 1.4.3.1 Can the task be offloaded : what to offload?

Unless Cloud data center workloads can be offloaded to edge devices and servers, edge computing promises to reduce service latency and network bandwidth usage cannot be fulfilled. As a result, in the coexisting edge-Cloud environment, original workloads conducted in Cloud data centers must be partitioned and some of them must be selected to be executed on edge devices. Furthermore, in other circumstances, such as the IoT, local computing power is dispersed heterogeneously among numerous heterogeneous devices, and local computing resources are insufficient to execute complicated programs. Therefore, careful selection of workloads that are offloaded to edge devices can result in delays and poor system performance [13].

#### 1.4.3.2 When to offload the task

Offloading computation allows the use of devices' computation storage, networking, and energy capabilities while lowering latency for compute-intensive applications and services.

Due to the fact that the network status changes dynamically during application execution, the decision to offload the load must indicate when it is to be offloaded [13].

#### 1.4.3.3 Where to offload

The task offloading can be implemented by scheduling partitioned tasks to the target edge device and edge server. The choice of edge device and edge server to target includes several objective optimizations such as performance, power, network bandwidth, privacy protection methodology. For example, an intrinsic scheduling policy is energy intensive tasks are offloaded to the Cloud servers to save energy, while data-intensive tasks are offloaded to the edge servers to reduce latency and network traffic [13].

#### 1.4.3.4 Which offload policy apply

A good computation offloading policy must find the optimal balance between the overall computation delay, data transmission, and related performance metrics [13]. A well designed task offloading strategy can save energy on devices and reduce the response time of applications [18]. For example, Lin et al. in [19] proposed an offloading method with task dependency on different processors based on Heterogeneous Earliest Finish Time (HEFT). Maio et al. in [20]. propose a heuristic-based approach to find a tradeoff solution among application runtime, battery lifetime, and user cost.

#### 1.4.3.5 What about trust (security)

Task offloading in MEC and Cloud computing are a solution to enhance resource-limited mobile devices' capabilities by offloading tasks to the their servers. The MEC/ Cloud computing may charge the mobile device for the calculation completed, so delivering offloading as a service. The difficulty here is how the device can trust that the MEC/ Cloud computing executes the expected computation and does not cheat [21]. Therefore, the realization of a secure and trusted resource sharing computing environment is one of the key elements of the development of edge computing [22].

## 1.5 Conclusion

In this chapter, we presented an overview of computation offloading in MEC. In the first and second sections, we presented an overview of Cloud computing and mobile edge computing, highlighting its advantages and benefits such as lowering response time and enabling mobile devices to perform their tasks by offering computational and storage capacity. We concluded these two parts with a comparison between the MEC and Cloud computing. Moreover, we

discussed computation offloading, which has been a cost-effective idea in recent times, as well as future AI-related applications that are resource restricted, etc. Finally, we presented certain questions to be addressed before to offloading, one of which being trust and security problems related to offloading and the importance of secure offloading to the Cloud/edge.

In the next chapter, we present a brief survey on blockchain and trust management systems for computation offloading in mobile edge computing.

CHAPTER 2

# When trust management meets blockchain

## 2.1 Introduction

Trust is one of the important issues for computation offloading in Cloud or edge computing. However, over the last several years, trust management approaches have been widely employed to identify and isolate untrustworthy behavior and objects. These strategies, however, still have several drawbacks. There is a need to protect the integrity and trustworthiness of the information that is transmitted and shared [23].

Blockchain (BC) is a novel approach to the problem of trust, fundamentally changing the way online transactions are conducted by securing the trust of unknown parties [24]. It uses encryption to protect users' identities, ensure secure transaction completion, and secure stored information. The combination of trust management with blockchain can provide secure data, more efficient verification of information integrity, and assist improve data confidentiality and availability while sharing and storing.

In this chapter, we start by presenting an overview of trust management in section 2.2 before presenting the blockchain technology in section 2.3. In section 2.4, we conclude with a discussion of blockchain-based trust management.

## 2.2 Trust management

Trust is an essential concept in all aspects of human contact. It enables us to interact with other people in the most secure manner possible, even when we lack all of the necessary communication or technology knowledge. In many cases, trust serves as a natural layer of protection inside human groups.

In this section, we give an overview on the key characteristics of the concept of trust management.

### 2.2.1 Definition

Trust management has been explored and used in a variety of domains, which include the social sciences, economics and philosophy, etc. As a result, we have found different definitions of trust management; below are some examples:

- According to *Tilborg et al.* "Trust management refers to the process of deciding whether the execution of a requested action is authorized by the combination of a local security policy and digitally signed assertions issued by trusted remote parties" [25].
- *Grandison and Sloman* define trust management as "The activity of collecting, codifying, analyzing and evaluating evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships" [26].

From these points, trust management is an effective method for assessing and establishing trusted relationships through the use of symbolic representations of social trust to facilitate automated decision-making.

### 2.2.2 Composition

According to Cristiano Castelfranchi, trust can be composed In several sub-beliefs [27]:

- **Competence belief:** one entity believes that the other entity is capable of doing what it expects from it in an efficient manner.
- **Willingness belief:** when one entity thinks that the other entity wants or needs to do what is expected of it in order to achieve its objectives.
- **Dependence belief:** An entity that is reliant on another because it believes it must rely on another entity to achieve its objectives since it cannot accomplish it alone.

### 2.2.3 Types

There is three types of trust [27]:

- **Blind trust:** This represents the "default" trust in the system before any event occurs, enabling entities to form interactions with objects that are unknown.
- **Conditional trust:** This is the "classic" trust state throughout the entite's whole lifetime. This conditional trust is likely to change and may be subject to limitations or requirements.
- **Unconditional trust:** A trust is most usually configured by the administrator and does not rely on successful or unsuccessful interactions, external suggestions, or other sources of conditional trust adjustments.

### 2.2.4 Key concepts

In the concept of trust management we find [28]:

- **Trustor:** The trustor is the person who establishes trust, which can be the cloud or an organization, for example.
- **Trustee:** The entity that asks access to the trustee's services is represented by an identifiable entity in the system. The trustees are autonomous in the idea that their activities can not be directly controlled by outsiders such as the service provider, hence the trustee is in charge of the trust's management.
- **Trust decision:** The trustee's decision is binary and based on the trust/risk balance.
- **Actions:** Use of the trustor's services is included. It may be necessary to lower the resources available at the time of the action or to strengthen the trustee's control.
- **Risk:** Any decision to enable exposes somethings under our control to aggressiveness or manipulation, while less monitoring means bad conduct goes undiscovered and more resources granted means they are wasted or exploited.
- **Reputation:** Is described as a predesigned notion generated by a party's past behavior regarding its intentions and conventions.
- **Recommendation:** It is the process of transferring a party's reputation from one communal environment to another.

### 2.2.5 Properties of trust relationships

Trust can be divided into the following categories based on different classification criterias [29].

- Direct trust and indirect trust.

  1) **Direct trust:** Is a trust relationship formed by the direct interactions of two entities. It is the consequence of exchanges between the trustor and the trustee.
  2) **Indirect trust:** In indirect trust, also known as derived trust or recommendation trust, the two people do not know one other. Trust is founded as a result of trustee intermediary individuals.

- Identity trust and behavior trust.

- Function trust and experience trust.

- Objective trust and subjective trust.

- Intra-domain trust and inter-domain trust.

- etc.

### 2.2.6 Trust management in Mobile edge computing

Trust management is an important part of the security mechanism in mobile edge computing environments, starting with analyzing the security and privacy of user data. Therefore, It is essential to build a trust evaluation scheme to evaluate service providers' performance in the MEC environment for four main reasons [30]:

- A service consumer knows the level of trust of a service provider before interacting with it.

- Service providers will know that their activities are tracked and logged in a database.

- The trust evaluation scheme allows a service provider to know its faulty points to improve them. Providers with good trust value will attract more service users, which increases their profits.

## 2.3 Blockchain technology

One of the issues that Internet users confront is making transactions in situations when they do not know or trust the other party. The blockchain is a novel technology that is being used to validate and preserve transaction records.

### 2.3.1 History

It all began in 1991, when researchers Stuart Haber and W. Scott Stornetta proposed a computationally feasible approach for time-Stamping digital documents so that they could

not be backdated or faked. Merkle trees were added into the design in 1992, making it more efficient by allowing several documents to be grouped into a single block [31].

In 2004, Harold Thomas Finney known as Hal Finney proposed the RPoW (Reusable Proof of Work) system, which might be considered an early prototype and an essential early step in the creation of cryptocurrencies [31].

Blockchain technology first emerged in late 2008, with the launch of the cryptocurrency bitcoin, when an individual (or group) writing under the pen name Satoshi Nakamoto posted a white paper titled « Bitcoin: A Peer to Peer Electronic Cash System » [31].

The Bitcoin cryptocurrency blockchain network was established in 2009. The pattern that most contemporary cryptocurrency schemes follow is outlined in Nakamoto's paper [31].

Ethereum was developed in 2013 by Vitalik Buterin, a programmer and creator of the Bitcoin magazine, with the intention of establishing a more adaptable blockchain that was not confined to currency transactions. In 2015, he introduced Ethereum as a second public blockchain capable of recording contracts, loans, and other types of transactions, whereas Bitcoin can only record transactions [31].

Nowadays, blockchain technology is gaining popularity and is employed in a variety of applications other than cryptocurrency.

### 2.3.2 Definition

Numerous definitions of blockchain have been given, including the following:

- A Blockchain is a distributed transactional database, comparable to a decentralized and shared ledger, that stores and transmits value or data via the Internet in a transparent, safe, and self-sufficient way without the intervention of a central control body [32].
- A blockchain is literally a chain of blocks, see figure 2.1, digital containers on which all kinds of information are stored: transactions, contracts, property titles, works of art, etc. The blocks constitute a database, much like the pages of a ledger. This ledger is distributed [33].
- Blockchain is an incorruptible digital record of economic transactions that can be configured to record not only financial transactions, but also nearly everything of value [34].

We can also propose this definition as a recapitulation of what precedes: Blockchain is a digital system that records transactions and asset data in several locations at the same time. In other words, blockchain is a technology that allows data to be stored and exchanged in a transparent, safe, and decentralized way, without the need for a central controlling authority .

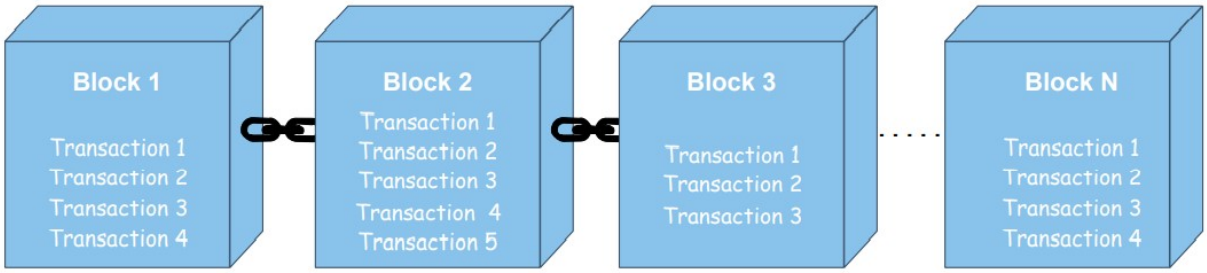


Figure 2.1: Simplified example of a Blockchain.

### 2.3.3 Main characteristics

Blockchain technology is mainly characterized by the following elements [2]:

- **Traceability:** The ability to track and validate the spatial and temporal information of a data block stored on the blockchain. Each block of data stored on the blockchain has a historical timestamp that guarantees data traceability.
- **Transparency:** The blockchain data is widely open to any user who can access and verify transactions on the chain. So, information can be verified by all stakeholders and accepted by consensus.
- **Anonymity:** when transferring data between nodes, the identity of the individual remains anonymous, making the system more secure.
- **Decentralization:** In blockchain, decentralization refers to the transfer of monitoring and decision making from a centralized association (individual, company, or group of people) to a decentralized network. Decentralized networks seek to reduce the level of trust that members should put in each other and the ability to exercise authority or direct each other in ways that undermine the effectiveness of the network.
- **Security:** Blockchain data is stored by a community of users rather than a single server, making it harder to destroy all copies of documents.

### 2.3.4 Components

The Blockchain contains numerous components that collaborate for in the completion of a transaction. The following are the key components of the Blockchain architecture:

- **Blocks:** Blocks are data structures designed to collect group of transactions and distribute them to all network nodes. Each block includes a header, which is the metadata that confirms its legitimacy, as well as the transactions that the miner has decided to include in the block that he or she has constructed. The maximum number of transactions that a block can contain is determined by the block's size and the size of each transaction.
- **Ledger:** A ledger is a store of current and previous state data maintained by all nodes in the network. There are three types of ledger [31]:
  1) **Public ledger:** It's open and transparent to everyone. Anyone on the blockchain network can read and write anything.
  2) **Distributed ledger:** Every node has a local copy of the database. Here, a group of nodes collectively execute the task.
  3) **Decentralized ledger:** the node or group of nodes does not have central control. Each node participates in the execution of the task.
- **Smart contracts:** Smart contracts are lines of code that are stored on the ledger of the blockchain and executed automatically when predetermined conditions are achieved [35].
- **Peer Network:** Peer-to-peer networks are used in blockchain networks where every node connects to every other node and shares resources with each other without the intervention of third-party authenticators or servers. All nodes have equal rights in applying the blockchain network [35].
- **Miners:** miners are some network nodes that allocate resources to verifying transactions and ensuring the blockchain's security. They are compensated through block rewards. A block reward is given to a single miner or a small group of miners for each new block [36].
- **Mining:** Mining is the method through which transactions are validated and added to the public ledger by miners [31].
- **Wallet:** this component is used to securely and privately store user credentials. It can store private keys, public keys, and associated addresses [2].
- **Membership:** It is used to provide an identity for a user to execute a transaction on a blockchain network [2].
- **Events:** It is used to generate notifications about transactions performed on the blockchain [35].
- **System management:** it used to create, monitor, or update the blockchain components [35].
- **System integration:** It is used to link the blockchain with the external systems [35].

### 2.3.5 Architecture

Blockchain technology can be divided into five layers: hardware and infrastructure layer, consensus layer, data layer, network layer, and application layer. The following figure illustrates the layered architecture of blockchain [37].

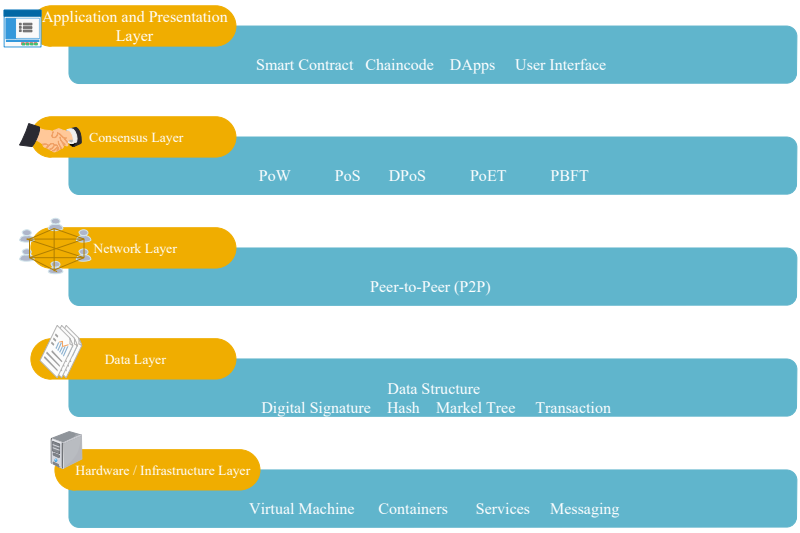


Figure 2.2: Layered blockchain Architecture..

1. **Application layer:** The application layer is composed of smart contracts, chaincode, and applications. It contains the applications that end-users utilize to engage with the blockchain network.

2. **Hardware and infrastructure layer:** Blockchain data is securely stored on the data server. When you browse the Internet or use a blockchain application, the machine requests access to this data from the server. The framework that facilitates this data exchange is called the Client Server model. Blockchain is a peer-to-peer (P2P) network that allows clients to interact with "peer-to-peer" connections.

3. **Data layer:** The blockchain data structure can be represented as a linked list of blocks in which transactions are ordered. The blockchain data structure contains two main components: pointers and linked lists. A pointer is a variable that points to the location of another variable, a linked list is a list of linked blocks, and each block contains data and a pointer to the previous block.

4. **Network layer:** The network layer, also known as the P2P layer, is responsible for communication between nodes. Handles discovery, transactions, and block propagation.

This layer is sometimes called the diffusion layer. This P2P layer allows nodes to discover and communicate with each other, propagate and synchronize to maintain a valid current state of the blockchain network.

5. **Consensus layer:** The consensus layer is the most important layer for the blockchain (Ethereum, Hyperledger, or any other). The consensus is responsible for verifying the blocks, ordering the blocks, and ensuring that everyone agrees.

### 2.3.6 Operating mode

The blockchain works decentrally without central control. Figure 2.3 describes a use case example to show the mechanism by which transactions work in a blockchain network. The details of this use-case are detailed in the following steps [2]:

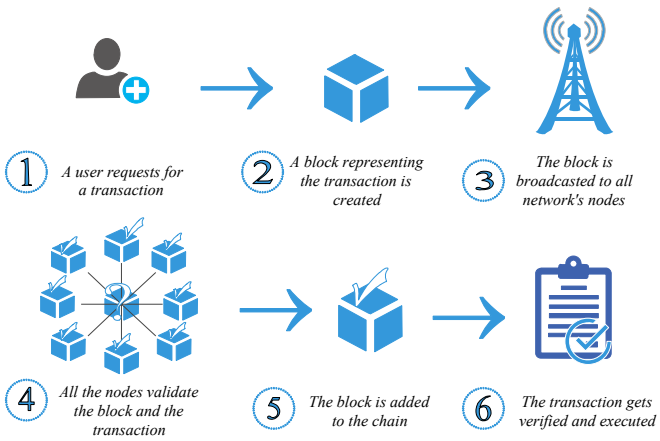


Figure 2.3: An illustrative scenario of how the Blockchain works [2].

1. A user A wants to make a transaction with B.
2. The transaction is added to a block. Thus, it can only be executed if all other network nodes verify it as a valid transaction.
3. The transaction is broadcast on the Blockchain network.
4. All the nodes validate the transaction.
5. The block is added to the existing blockchain, and the update is distributed across the network.

6. Finally, the transaction is confirmed and B receives the transaction that is sent from A.

### 2.3.7 Types

There are four types of blockchain, which are as follows [31] :

#### 2.3.7.1 Public blockchain (permissionless blockchain)

Everyone in these blockchain systems can contribute to the network's consensus mechanism. In addition, anybody with an internet connection anywhere in the world may transact and examine the whole transaction history.

- **Advantages**
  - Public blockchains are largely secure as long as the security rules are respected.
  - Network members do not need to trust each other for transactions to be executed and secured.
- **Disadvantages**
  - Processing is rather slow.
  - A block's transactions take a long time and require a lot of energy.
- **Exemples**
  - Bitcoin, Litecoin, Ethereum, etc.

#### 2.3.7.2 Private blockchain (permissioned blockchain)

These blockchain systems limit the ability to transact and read the transaction log to just the system's participating nodes, and the architect or owner of the blockchain system has control over who may engage in the blockchain system and which nodes can participate in the consensus process.

- **Advantages**
  - It quickly processes a large number of transactions for each block.
  - Permission levels, security, authorizations, and accessibility are determined by the governing organization.
- **Disadvantages**
  - The limited number of nodes may also indicate a lack of security.

- On a private blockchain, there is no anonymity.

- **Exemples**
  - Rubix, Hyperledger, Passcare, etc.

#### 2.3.7.3 Consortium blockchain

Consortium blockchain or federated blockchain works in the same way as a private blockchain, with access restricted to a particular group. In this type of blockchain, a validator node can initiate, receive, and validate transactions. However, the transactions can be received and launched only by member nodes.

- **Advantages**
  - It Access control ;
  - Scalability and security.
- **Disadvantages**
  - Consortium blockchain is less transparent than public blockchain.
- **Exemples**
  - Ripple, R3.

#### 2.3.7.4 Hybrid blockchain

Hybrid blockchains are characterized by the creation of a private system combined with a public system. This gives you control over who has access to certain data stored on the blockchain and what data is exposed to all network participants.

- **Advantages**
  - Access control.
  - Performance and scalability.
- **Disadvantages**
  - Isn't completely transparent;
  - Upgrading.
- **Exemples**
  - XinFin, etc.

### 2.3.8 Consensus models

The blockchain is a peer-to-peer network and there is no central node that guarantees that the ledgers of the distributed nodes are all the same. To ensure that all nodes agree on the transaction and the order listed in the validated block, the blockchain uses a consensus protocol that employs a variety of methods. There are several consensus protocols. The most well-known ones are listed below [36].

#### 2.3.8.1 PoW (Proof of Work)

Proof of work is a consensus mechanism used to confirm transactions and generate new blocks in the Bitcoin network, as well as many other cryptocurrency networks. It is the most often used Blockchain consensus protocol.

In the proof-of-work model, miners in the network compete with each other to solve complex computational puzzles. These puzzles require a high level of computational power and are difficult to solve, but it is easy to verify the correctness of the solution. When a miner finds a solution to the puzzle, it can broadcast the block to the network, and all other miners can verify the correctness of the solution. As a reward for verifying these transactions, miners are paid cryptocurrency from the network.

PoW effectively secures the network, making hacking attempts very difficult, but the disadvantage is that it necessitates the use of powerful computation resources that consumes a lot of power.

#### 2.3.8.2 PoS (Proof of Stake)

Proof of stake is a similar method to proof of work, but the process is different. Rather than proof of work, in which the algorithm rewards miners who solve a mathematical problem that results in the creation of a new block, the creator of the new block is selected from a pool of users who staked a particular amount of cryptocurrency.

This indicates that there is no puzzle to solve in the proof of stack system, so there is no reward for doing instead of the minner taking a feat from every transaction; this also means that because nobody is competing to solve there's no massive energy requirement.

Because the system's security is weak, the system's simplicity may result in an attack of the type "Nothing at stake," which may be defined as "Nothing to lose."

#### 2.3.8.3 DPoS (Delegated Proof of Stake)

DPoS is a consensus mechanism where delegates are voted in by coin holders responsible for maintaining the ledger, adding blocks, and generating new coins.

To begin, holders perform a unique transaction that provides influence to a delegate, which is sometimes referred to as a computer or server engaging in consensus. The amount of power they have is based on the number of coins they own. The coin holders do not fund delegates; instead, they distribute power. This is known as voting, and the network then adds up all of the delegates' influence to determine which delegates are allowed to participate in consensus. Authorized delegates fill the top positions for a set length of time. And at last, authorized delegates convene to validate transactions, add new blocks to the chain, and generate new currencies.

DPoS has the advantage of reducing interactions between nodes and allowing for a bigger number of transactions and fastest validations. The primary drawback is increased centralization, which increases the danger of vulnerability, censorship, and cartelization .

### 2.3.9 Challenges

The challenges of blockchain technology include the following [36], [38]:

- **Security:** Like any other technology, the blockchain comes with some security issues. Transaction address anonymization still cannot guarantee user anonymity, and some deliberate attacks have been shown to still pose a threat. Related transactions between different addresses can expose personal information, which is fatal for commercial use. Moreover, with the development of quantum computing, encryption itself is also under threat. Therefore, technology needs to be constantly updated to ensure the security of the blockchain.
- **Latency:** The blockchain takes more time to complete a transaction. Therefore, it slows down the process and requires more computing power.
- **Storage issues:** In the blockchain, all transactions are recorded in blocks, and by protecting different nodes in a distributed system, you can effectively prevent the ledger from being tampered with. However, with the accumulation of transactions as the data grows, the remaining space in each block gets smaller and smaller.
- **Cost issues:** Blockchain relies on encryption to provide security and establish consensus on distributed networks. This means that writing to the chain requires a complex algorithm aimed at empowering the user. It is cheaper to run this type of complex algorithm because it requires more power.

### 2.3.10 BC-based trust management

Typically, a trust management system is required to respond based on real-time evaluations of entities' conduct during established connections, as well as comments and recommendations gathered from other entities. However, at a time when there is a crisis of faith and trust in traditional third parties, Blockchain technology, which offers transparency, both seduces and fascinates.

As result, BC-based trust management system that enables transparent, traceable, and autonomous data, software, and infrastructure management and transactions between trusted entities and partners. Several ways have been given in recent years, but this is the most recent trend in generating decentralized and distributed trust. This paves the way for the successful deployment and operation of intelligent applications with complete trust along the whole edge-to-Cloud continuity. BC-based trust management systems have been applied in a range of scenarios, including IoT, cloud computing, E-Commerce, automotive networks, etc.

Liao et al. in [39] created a safe and intelligent task offloading system to handle the difficulties of minimizing task offloading time, queuing delay, and handover cost with partial information while maintaining privacy, fairness, and security. They use blockchain and smart contracts to provide equitable job offloading and to reduce different security threats. Nguyenet al. in [40], addressed the security and compute offloading issues in a mobile edge-cloud computation offloading (MECCO) system with blockchain at the same time. To begin, in order to improve offloading security, they propose a trustworthy access control system based on blockchain that can secure cloud resources from illicit offloading behavior. Zhu et al. in [41], enhanced Proof-of-Trust (PoT) consensus is presented with blockchain as the underlying technology, which is appropriate for crowdsourced service applications. This PoT consensus uses a subjective logic reputation process to identify nodes with high trust. Lahbib et al. in [23], proposed a safe trust management system based on blockchain technology to benefit from the secure sharing and storage of trust information.

In table 2.1 , we provide a taxonomic classification and recapitulation of the previous cited works on BC-based trust management. These citations are classified based on five key characteristics.

| Works | Cloud/ Fog /Edge | Blockchain | Trust management | Security | Incentive |
|---|---|---|---|---|---|
| Liao et al. (2021) [39] | Fog | ✓ | ✓ | ✓ | × |
| Nguyen et al. (2021) [40] | Cloud | ✓ | × | ✓ | × |
| Zhu et al. (2020) [41] | - | ✓ | ✓ | × | ✓ |
| Lahbib et al. (2019) [23] | Cloud | ✓ | ✓ | × | × |
| **Our scheme** | **all** | ✓ | ✓ | ✓ | ✓ |

Table 2.1: Taxonomic recapitulating summary of existing related works.

**Discussion :** As seen the blockchain technology has been used for several purposes and within various fields and application domains. However just a few studies [23], [40] have focused on integrating the blockchain technology within trust and reputation systems. existing works on BC-based trust management consider only reputation-based trust in one direction of the interaction between trustee and trustor. However, trust must be established in the two directions of an interaction. Also almost all existing works investigate the indirect trust and therefore make the trust decision based on the history using reputation score. Nevertheless, direct trust is very important. We can't trust an entity based only on the fact that it has a good past, but we have to also consider the present and evaluate the current interaction to make any decision.

## 2.4 Conclusion

In this chapter, we introduced trust management, an essential technology for performing transactions between two entities. We also present the Blockchain technology, a new revolutionary technology that allows users to perform transactions that are guaranteed and auditable by everyone without the need for a trusted third party. To conclude, we discussed blockchain-based trust management.

BC-based trust management system enables transparent, traceable, and autonomous data, software, and infrastructure management as well as transactions between trusted entities and partners. This sets the way for the effective deployment and operation of intelligent applications with total confidence along with the whole edge-to-cloud continuity.

Motivated by the previous discussion on the BC-based trust management systems, we present in the next chapter a Blockchain-based trust management system to secure task offloading in mobile edge computing.

CHAPTER 3

# BLOCKCHAIN-BASED TRUST MANAGEMENT SYSTEM FOR TASK OFFLOADING IN MEC

## 3.1 Introduction

As the Internet of Things advances, mobile devices are becoming increasingly prevalent. Mobile devices, on the other hand, lack the power and resources to execute applications, using huge amounts of energy, and require significant processing capacity as well as stringent time constraints.

Offloading computations from mobile devices to distant servers has recently been proposed as a way to improve mobile device application performance while using less energy. One of these servers is MEC, a new paradigm in which devices may communicate quickly with the nearest edge node, significantly reducing data latency. Unfortunately, because the two entities (MEC and mobile devices) do not know each other, several security trust concerns have arisen. With the emergence of the Blockchain, which is gaining popularity because of its autonomy, privacy, and immutability, as well as the spread of recorded data The Blockchain enables offloading to be secure, safe, dependable, transparent, and tamper-proof.

In this chapter, we propose a new blockchain-based trust management system to secure task offloading in mobile edge computing. In section 3.2, we formulate our main problem and describe the network model before moving to the system model. In section 3.3, we propose a secure trust management protocol for task computation. After that, in section 3.4, we describe the BC-based incentive mechanism for block verification. Following that, in section 3.5, we'll look at how to model this motivational strategy using a Stackelberg game and the best system parameters for this approach in section 3.6. Finally, in section 3.7, we conclude this chapter.

## 3.2 Problem formulation

Due to the limited computing capacity of the mobile devices, denoted $m_i$, where $m_i \in (m_1, m_2, \ldots, m_n)$, they to offload their computation-intensive tasks to the MEC server, denoted $s_i$, where $S_i \in (s_1, s_2, \ldots, S_n)$. MEC servers have more powerful computing and storage to accomplish more efficiently intensive tasks, denoted $t_i$, where $t_i \in (t_1, t_2, \ldots, t_n)$.

The absence of confidence between mobile terminals and MEC servers presents a massive security and privacy problem to the implementation of the computation/data offloading technique. Indeed, before engaging in any interaction, each party in the system must check and assure the other party's trustworthiness and security. An architecture with a third party that assures trust [42] is one of the options presented to eliminate the trust problem, although this would centralize the system and not totally tackle the trust issues.

Motivated by the above discussion , we propose to use blockchain (BC) technology to secure trust management of computation offloading in MEC. Each system user will have a copy of the blockchain, and interactions will take place directly and anonymously without the need for a third party to ensure trust. Each block will contain information such as the hash of the block as well as the reputation score of the system entity, and based on the stored reputation the trust will be calculated.

### 3.2.1 Network Model

As shown in figure 3.1, we consider an integrated edge-cloud architecture for IoT networks that consists of three fundamental layers: mobile devices, edge computing, and cloud computing. The key components of each layer are detailed below [40].

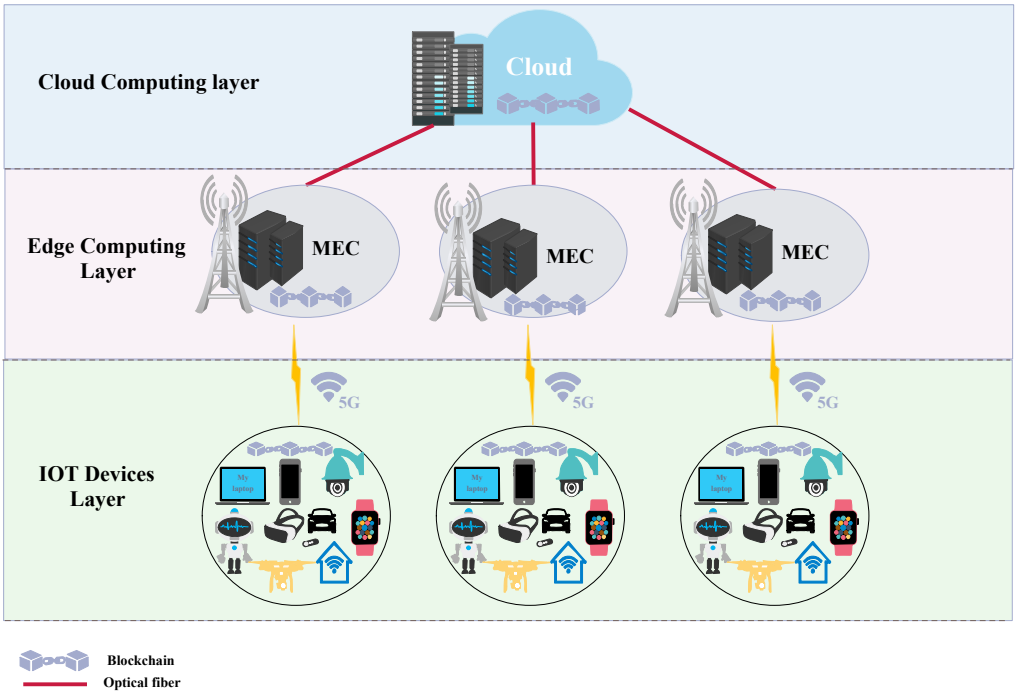


Figure 3.1: The network architecture of the BC-based computation offloading trust management system.

1) **IoT devices layer:** This layer contains up the IoT mobile devices responsible for the collection and the analysis of data. Sensors, drones, and smartphones are examples of these devices. To make better use of the system, these devices are classified based on their geographic location. [40].

2) **Edge computing layer:** This layer is composed of wireless access points or base stations that provide wireless communication to nearby devices and edge nodes. This layer, in particular, provides calculation, verification, and analysis of reliability-related data, allowing IoT devices to offload their computation-intensive tasks [40].

3) **Cloud computing layer:** This layer consists of a cloud server that is considered as a Trusted Authority (TA). This layer is in charge of providing security settings to IoT devices such as keys.

The wired technology of optical fiber and wireless of 5G are used to connect the different layers. Indeed, optical fiber technology is utilized to connect the components of the edge computing layer to those of the cloud computing layer. This is because it is the best telecommunication technology capable of delivering thousands of billions of bits per second across large distances while still offering large capacity. Also, 5G is the latest evolution of cellular networks that has 100 times faster connectivity rates, ultra-low latency, and higher capacity than prior gener-

ations. Both 5G and edge computing have the potential to significantly increase application performance and enable the real-time processing of massive volumes of data.

### 3.2.2 System model

We consider a blockchain-based trust management system for task offloading in MEC. The main purpose of this work is to provide a secure and reliable interaction between MEC and IoT devices that do not know or trust each other. Blockchain technology underpins this system. Our system is composed of six main entities which are: trust authority, computation task requester, task computing agent, verifiers, auditors, and the blockchain, see figure 3.2 Their roles are described below :

(a) **Trust Authority (TA):** a trustworthy entity that may be a cloud provider (e.g., Amazon AWS, Microsoft Azure, Google GCP, etc.). It is responsible for system initialization, public parameter generation, user registration, and the generation of secret and public keys for each user.

(b) **Computation task requester (CTR):** they are vehicles, smartphones, drones, and other IoT devices with limited resources. These mobile devices frequently execute computationally complex AI-based applications such as augmented virtual reality, image processing, object identification, etc. Due to their limited computational and energy resources, they are often forced to offload and request the computation of their tasks remotely on edge/cloud servers under an access control mechanism for security assurance.

(c) **Task computing agent (TCA):** it represents the edge servers that receive the computation offloading requests from the computation task requester. They may collaborate to compute the offloaded tasks.

(d) **Verifiers:** they can be edge servers, mobile IoT devices, or any entity in the system that has a good reputation and a copy of the blockchain. In exchange for a set of tokens as an incentive, these verifiers participate in the block verification and therefore in the consensus process.

(e) **Auditors or validators:** They are edge servers or mobile IoT devices that exist in the system. Verifiers encourage auditors to validate their verification by signing a smart contract in exchange for a number of tokens. The more auditors who sign the contract, the more reliable the verification will be.

(f) **Blockchain:** Since blockchain-based networks are open and transparent and show promise for the collection of credit data with good attributes of tamper resistance and decentralization, we have integrated blockchain into our system in order to guarantee trust between the different components of the system.

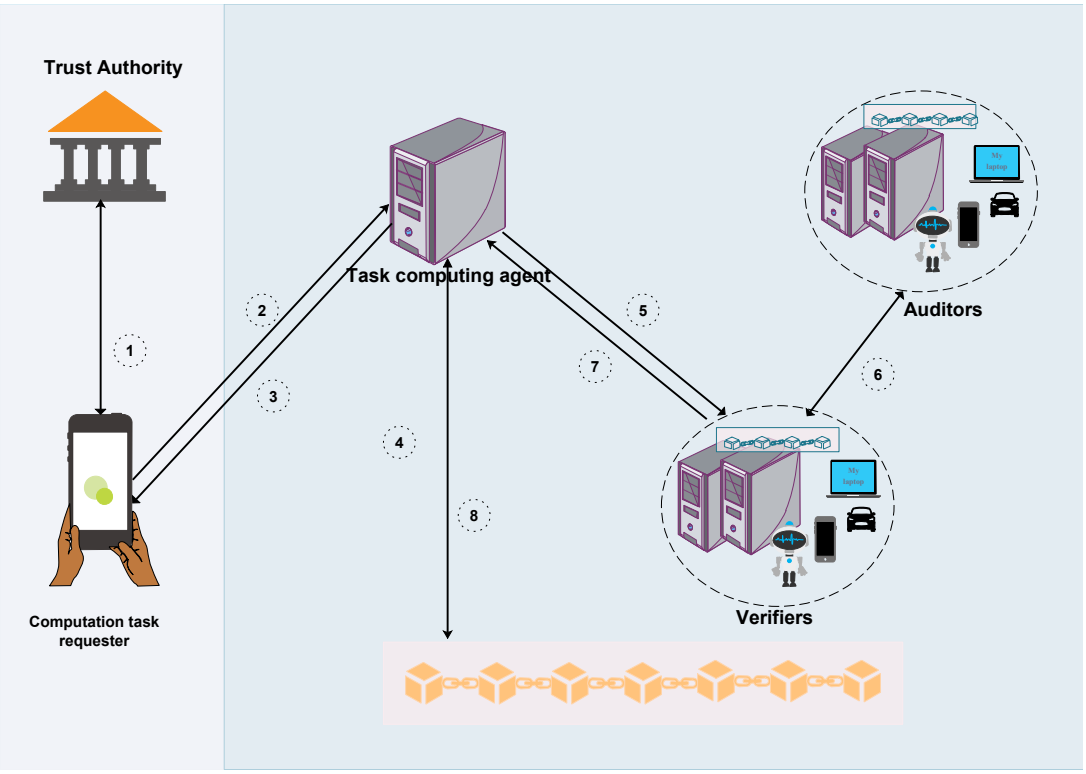


Figure 3.2: The architecture of the proposed BC-based trust management system.

## 3.3 A secure trust management protocol for task offloading in MEC

In this section, we present our proposed BC-based trust management protocol to secure task computation in MEC. We start by describing the operating mode of the protocol. After, we detailed its operational phases.

Before detailing the phases of our BC-based trust system, we start by describing the different packets used to manage the blockchain.

#### 3.3.0.1 Type of packets

The main type of packets used by the BC-based trust management system for task offloading are detailed as follow:

- **Request offloading packet (ROP):** ROP refers to a type of request packet generated by a computation task requester and sent to the TCA. The packet provides information on the offload request, such as its masked ID, task complexity in terms of CPU cycles, size of task data, and a threshold that represents the task's computation deadline to receive results.
- **Request verification packet (RVP):** the TCA generates the RVP and sends it to the most

efficient (the nearness and/ or the most trustworthy for example) verifiers to verify the computation task requester's candidate block.

- **Request verification response packets (RVRP):** They are created by the verifiers and broadcast to the auditors or validators for the validation of their block verification.
- **Reply audit packets (RAP):** They are created by auditors in order to respond to verifiers. If a trustworthy auditor decides to favorably react to the RVRP packet, it adds his digital signature and sends it back to the verifier.
- **Announcements-aggregated packets (AGP):** AGP refers to the type of aggregate packet generated by the verifiers and sent to TCA, including the request and signature information for all participating auditors.

#### 3.3.0.2 Operating Phases

The operating mode of our proposed BC-based trust management for task offloading system can be organized into eight major phases as illustrated in 3.2. These steps are slightly inspired by similar works in [43], [44] and are briefly detailed below:

The initialization phase consists, we have two phases as explained below.

1) **Initialization:** The trust authority uses the elliptic curve to register, generate, and assign a pair of keys (private and public) to each new system user. These keys will be used to encrypt messages in the future. A new user could not be legitimate until after receiving its pair of keys and being approved by the TA.
   (a) **Setup phase:** at the start of the system, the trust authority defined a set of data points G on an elliptic curve. G's generating function is represented by P. H is also defined as a hash function. Assuming there are n mobile terminal users, it will do the following actions [45]:
      i. For the $n$ mobile users, it generates $n256$ random bytes strings $(k_1, k_2, \ldots, k_n)$ and based on computes $k_i$, where $k_1$ $(k_1, k_2, \ldots, k_n)$.
      ii. SHA-3 is utilized as the hash function H.
      iii. The master private-key vector is defined as a collection of private keys X = $(k_1, k_2, \ldots, k_n)$.
      iv. The master public-key vector is defined as a collection of public keys $Y = (K_1, K_2, \ldots, K_n)$.
      v. Finally, the trust authority publishes and broadcasts (G, Y, H) as system public parameters.
   (b) **Registration:** After obtaining its public key, the mobile device will perform the following actions :
      i. To create his identifiant $ID_i$, the device first generates a random number of 20 octets which is then appended to his IPv6 address.
      ii. It will then compute and save its masked identity.

2) **Offloading request formulation:** Because of resource constraints of IoT mobile devices, the computation task requester (mobile device $i(i \in M)$ ) sends a computation offload request to the nearest TCA $j(j \in S)$, this task is indicated as $ROP$. The offloading request includes task information such as task data size, maximum task delay tolerance for completion, task complexity in terms of CPU cycles, disguised identity, and public key. To generate $ROP$, the mobile device must follow the next steps:

   (a) Generates a random one-time private key $Pkq_i$ and uses it to compute the associated public $Pkp_i = Pkq_i \times G$. Moreover, it adds its $MID_i$ .

   (b) $D_i$ (in bits) specifies the data size of the computation task of device i [40].

   (c) The total number of CPU cycles necessary to complete the computation of task i is defined as $\lambda_i$ (in CPU cycles/bit). Furthermore, $\tau_i$ (in seconds) denotes the maximum tolerated computation completion delay (response time) of the task [40].

$$ROP = \{D_i, \lambda_i, \tau_i, MID_i, Pkp\}. \tag{1}$$

3) **Direct trust score computation:** When the TCA receives an offload request, it computes the direct trust score (DT) between itself and the computation task requester and decides the trust level based on the users' opinions. The TCA i uses subjective logic to define its opinion on the computation task requester j which is called local opinion. We use $b$ for belief, $d$ for distrust, and $u$ for uncertainty [46], where :

$$b_{i\to j} + d_{i\to j} + u_{i\to j} = 1 \quad \text{and} \quad b_{i\to j}, d_{i\to j}, u_{i\to j} \in [0,1] \tag{2}$$

where $a_{i\to j}$ is a base rate $\in [0,1]$. According to the subjective logic model [47] :

$$\begin{aligned} b_{i\to j} &= \left(1 - u_{i\to j}\right) \frac{r_i}{r_i + s_i} \\ d_{i\to j} &= \left(1 - u_{i\to j}\right) \frac{s_i}{r_i + s_i} \\ u_{i\to j} &= 1 - q_{i\to j} \end{aligned} \tag{3}$$

where $r_i$ is the number of correct transactions, $s_i$ is the number of incorrect transactions, and $q_i$ is the quality of the connection between the computation task requester and the task computing agent, which is linked to the packet's successful transfer rate. Then the direct trust score is calculated as follows [46], [48]:

$$DT_{ij}(t) = b_{i\to j} + a_{i\to j^*} u_{i\to j^-} \tag{4}$$

4) **Reputation-based indirect trust score:** Once the direct trust has been determined, the indirect trust value is obtained from the blockchain. This value is based on a reputation score (RS) detained by each user and stored on the blockchain. This score is continuously updated after each interaction of a user. At this stage, the TCA builds an initial block and sends a reputation score verification request to the verifiers. The reputation score is represented by a group of tokens or virtual coins.

   When the TCA obtains from the blockchain the reputation score RS of the computation task requester, the computation task agent prepares its $RVP$.

(a) TCA selects $\left(MID_{v_1}, MID_{v_2}, \ldots, \text{MID}_{v_L}\right)$ from the nearest verifiers with high reputation scores.

(b) Creates a random one-time private key $Pr_{TCA}$, calculates the matching public key $Pk_{TCA} = Pr_{TCA} \times G$,

(c) Generate the signature of the task computation agent.

$$\text{Sig}_{TCA} = H\left(MID_{TCA} + PK_{TCA}\right) \tag{5}$$

(d) Creates an anonymous message ($msg$) that contains the RS and $MID_i$,

(e) Choose the value of the threshold t and define $RVP$ as follow:

$$RVP = \left\{\left(MID_{v1}, MID_{v2}, \ldots, MID_{vL}\right), t, msg, \text{Sig}_{TCA}\right\} \tag{6}$$

(f) The TCA creates an initial candidate block that contains the trust value which is calculated based on a combination of direct trust score and indirect trust score.

5) **RS verification:** The verifiers will check the value of the reputation score supplied by the task computing agent. After checking RS in its own blockchain, each verifier will broadcast a message to a set of auditors to validate this RS and by signing a smart contract using its private key to validate its verification. As many of the signed contracts by auditors (validators) as higher will be the verification response of the verifier. Task computing agent sends $RVP$ to the verifiers to check the validity of the obtained reputation score.

(a) The verifier extracts from RVP the $PK_{TCA}$ of TCA.

(b) For the $MID_{TCA}$, the verifier $v_i$ verify the following equation :

$$\text{Sig}_v = H\left(MID_v + PK_v\right) \tag{7}$$

If the $sig_{TCA}$ signature does not satisfy the relation 7, $v_i$ will not accept the verification of the RVP of TCA. If it verifies condition 7, $v_i$ considers the RVP since the message's signature has been validated as legitimate.

(c) $v_i$ will retrieve the MIDi and reputation score from the msg and then verify the RS of the MIDi from the copy of the blockchain that it holds; if the RS determined by the task computation agent is correct, $v_i$ will broadcast $RVPs$ to the audits.

$$RVPr = \left\{\left(MID_{\text{audit } 1'} MID_{\text{aud it } 2}, \ldots, MID_{\text{audit } n}\right), msg, \text{Sig}_{v'}\right\}. \tag{8}$$

6) **RS audit or validation:** When auditors receive a request for auditing or validation from a verifier, they check the signatures and reputation score found in the request, compare their RS with that in the audit request, and then append their signature to the verifier's message and send it back to the verifier. After receiving RVPs, auditors will extracts from RVPr the $Pk_v$ of the verifier,

$$\text{Sig}_v = H\left(MID_v + PK_v\right) \tag{9}$$

When the signature satisfies the relation (9), the auditor validates the RS verified by the verifiers. Once RS is correct the auditor generates a message $msg^{'}$ as rQTR to the verifier Such as : $msg^{'} = msg + Sigau_{dit}$

7) **Verification response:** TCA receives the verification answer from the verifiers. The verifier submits his response after receiving all of the auditors' responses. It gathers all the responses and signatures of the auditors in an aggregate packet and sends it back to TCA. The verifier combines the rQTRs to generate an aggregate message (AGP) and send it back to the TCA such as:

$$AGP = \{< MID_{\text{audit }1}, msg'_1 >, \ldots, < M_{\text{audit }'}, m_1 g'_n >\} \tag{10}$$

8) **Add block:** When the number of successful verifications reaches the threshold ( of verifiers), the reputation score is considered valid and the total trust score will be derived by combining direct and indirect trust scores. Therefore the candidate block will be validated and added to the blockchain and the TCA can start the task computation. Otherwise, the offloading request will be considered malicious and it will be immediately dropped. When the task computing agent receives the threshold of rQTR, it performs the following steps [45]:

   (a) for each $MID_{auditi} \in (MID_{audit1}, MID_{audit2}, \ldots, MID_{auditn})$

   $$V = H(M_{\text{audit }i} + PK_{\text{audit }i}) \tag{11}$$

   Where $V$ is the signature that is used for non-repudiation of information and can prevent message falsification.

   (b) for each $MID_{auditi} \in (MID_{audit1}, MID_{audit2}, \ldots, MID_{auditn})$, the TCA compares the signature of each audit ($Sig_{auditi}$) to the signature calculated:

   $$\text{Sig}_i = H(MID_i + PK_i) \tag{12}$$

   if the couple <MIDaudit i, msg'i> satisfy the relation 12, the TCA considers the message; otherwise, the message is immediately dropped . The TCA proceeds to the next step. The TCA calculates total trust to get the overall final trust level [48].

   $$T_{ij}^{\text{total}}(t) = \omega \cdot DT_{ij}(t) + (1 - \omega) \cdot IT_{ij}(t). \tag{13}$$

   $\omega$ is the parameter that describes the contribution of direct and indirect trusts to total trust value.

### 3.3.1 Construction of blockchain

Our system uses Blockchain to store users' identities and reputation scores in a transparent and anonymous way. The blockchain is also used in the incentive mechanism to encourage users to participate more actively in the system.

As illustrated in figure 3.3. The structure of the block is as follows [45].

$B \equiv (B_h, B_t, B_u)$ , where :

- $B_h$, is the block header information.
- $B_t$, is the transaction within the block.
- $B_u$, is the block header list of the uncle node.
- $B_h$, is composed of : the block number (index of block), the time stamp, data, the previous block hash, and the block hash.

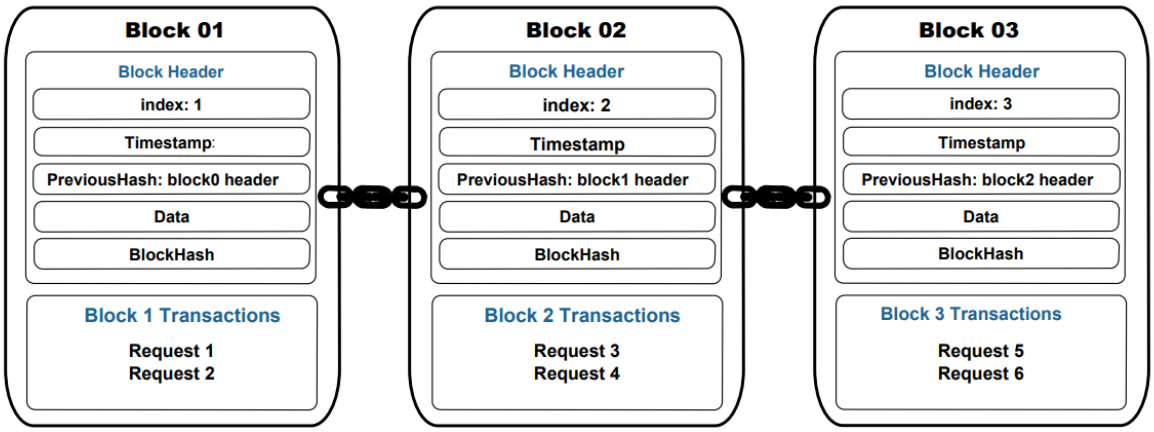


Figure 3.3: The generic blockchain.

The blockchain consists of a series of blocks, as illustrated in figure 3.3. In other words, the number of blocks on the master chain determines the depth of the blockchain and is denoted as follows [45]:

$$Blockchain := Block_1 || Block_2 || ... || Block_k$$

Upon reaching consensus, the block manager extracts the new block, records the transaction in the current block, and then adds the new block to the blockchain until another confirmation is made.

$$Blockchain := Block_1 || Block_2 || ... || Block_k || Block_{new}$$

We describe the consensus scheme that we adopt for the block, generation, verification, and validation.

### 3.3.2 DPoS-based Consensus scheme

PoS is presented as an improvement over PoW because it requires no hardware or energy consumption [49], and it's recently adopted by Ethereum [50].

In our system of trust management, we adopt DPoS as the consensus mechanism, which is an update of PoS. DPoS is faster than PoS, more decentralized, and consumes less power than PoS [51].

DPoS is suitable for secure data exchange between peers. To that end, the DPoS steps that apply to the incentive mechanism are detailed below [43]:

- **Step 1 -System initialization:** Any entity wishing to join or participate in the system must be certified by the trusted authority (TA) to become a legitimate legal entity. Each entity that participates in the system is given its own public key, and private key [45] following the steps detailed previously in the subsection (6). A system entity (e.g., edge server or device) that wants to be a candidate for a position as a miner first submits its identity information to the TA. The TA accepts only an entity as a miner candidate if the average reputation is above a confidence threshold [43].
- **Step 2 -Joining miner candidates:** Each miner candidate makes a deposit of stakes (tokens or virtual coins) in the account. If the candidate miner behaves maliciously and causes harm during the consensus process (e.g., the miner failed to create a block in its time slot) the blockchain system will confiscate this deposit and will be removed from the account [52], [43].
- **Step 3 -Reputation calculation:** Stakeholders can calculate the reputation of all miner's candidates. The reputation of each miner candidate is initially calculated based on the direct trust using subjective logic and the indirect trust using the consensus algorithm. This reputation is represented by the number of tokens that each mobile device has in its account (virtual wallet). These tokens increase and decrease after each participation in the trust management process and block verification. The mobile device receives tokens at the end of the consensus for each honest and non-falsified participation. However, for each fraudulent or malicious participation, it loses a portion of these tokens [43].
- **Step 4 -Miners selection:** After calculating the reputation, we choose as miners all the system entities that have a reputation score greater than or equal to that of a computation task requester. This means that edge devices with a number of tokens greater than or equal to the number of tokens of the computation task requester will be chosen as the set of active miners, and other edge devices will be considered standby miners.
- **Step 5 -Block generation by block manager:** The active miners will take turns acting as block managers during the N time slots of the consensus process. As in traditional DPoS consensus schemes, each active miner takes on the role of a block manager that creates, propagates, validates, and manages blocks in the time slot [43]. First, the computation task requester sends an offload request to the TCA. TCA will be the first block manager with a time slot equal to the minimum request reception time (predefined time period). The TCA transmits the block to the verifier, who becomes the new block manager with a time slot equal to the minimum time required to verify the block. The TCA will add the block to the

blockchain at the end of the verification period if the event appears to be credible, and the information will be broadcast to all system entities to update their copy of the blockchain.

- **Step 6 -Consensus process:** First, an unverified block will be generated by the TCA; after that, it will be distributed to other active miners for verification. Therefore, the number of active miners is limited, malicious active miners can launch block validation collision attacks and produce incorrect block verification results. In the phase of block verification, more verifiers and auditors lead to a more secure blockchain network [44]. Thus, the proposed DPoS consensus scheme is more secure when more verifiers participate in block verification, rather than just the active miners. This is because verifiers are incentivized to do so through rewards, which improves the security of the network. That is to say, Miners, both active and on standby, can function as verifiers and take part in the block validation process. In particular, miners with high reputation can prevent active miners from collusion in block verification [43].
- **Step 7 -Reputation update** Following the consensus process, if the block is validated by the verifiers, it will be added to the blockchain and the TCA wins the reward. Afterward, it will distribute this reward to all participants in the verification process. On the other hand, if the block is not validated, the participants (TCA, auditors and verifiers) lose their reputations that have already been submitted for the right to participate in the process. Moreover, because each verifier has a group of auditors that they assisted in verifying, the verifier must share a part of their profit with them. To encourage verifiers to verify and successfully engage in our BC-based trust management system, we propose a blockchain-based incentive mechanism. The following sections go through the main aspects of this incentive scheme.

## 3.4 BC-based incentive mechanism for block verification

To motivate the mobile devices and/or edge servers to participate efficiently in the block verification step of the blockchain consensus protocol, we propose a BC-based incentive mechanism. In this incentive mechanism, the TCA encourages the blockchain verifiers to participate in the verification/validation process of the block and therefore the calculation of the indirect trust score. TCA uses a part of its profit earned from the task computation as an incentive to encourage verifiers. A verifier can be any component of the blockchain system with a minimum of tokens, i.e., reputation.

In our incentive mechanism, short is the verification time, higher will be the earned incentive. Also, verifiers with high reputation scores will get larger rewards as an incentive. Furthermore, a verifier needs to validate (sign the smart contract) its response by a set of validators/auditors. Hence, the higher the number of validators, the greater will be the reward.

We propose a formal incentive model that uses the Stackelberg game as a modeling tool to encourage verifiers and auditors with a high reputation to participate in block validation

operations. Models based on the Stackelberg game are also used to share profits among verifiers and auditors. Our system employs a model based on the Stackelberg game to distribute profits among the verifiers based on their participation (the one who responds the fastest and with a good reputation receives higher profit). The game-theoretic-based incentive model is described in detail in the next section.

## 3.5 Stackelberg game-based incentive mechanism for block verification

We model our proposed BC-based incentive mechanism for trust management in MEC using a Stackelberg game (SG). Stackelberg game is a sequential game that is usually used to model and investigate incentive mechanisms between two types of competitive and rational players known as leaders and followers [53]. The leader starts first by announcing its strategy and then the followers compete with each other and reply with their best strategy that optimizes the leader's strategy. After receiving all the responses of the followers, the leader optimizes its final strategy. For more details on game theory and the Stackelberg game, refer to Annex A.

As illustrated in figure 3.4, our Stackelberg game is a two-stage game model played between two entities of the BC-based trust management system, which are:

- **Leader:** The leader is the task computing agent that initializes the game by announcing its initial strategy in the first stage. The leader strategy consists of the verification response deadline, denoted $\tau_j$, and the total amount of the incentive donated $\Re$ . The leader aims to minimize its utility, which is represented by its level of satisfaction regarding the follower's responses in terms of verification delay and verification cost.
- **Followers:** The followers in our SG are the block verifiers that represent the mobile devices or edge servers with a reputation score at least equal to the minimum tokens required for block verification. Indeed, after receiving the initial announcement of the leader, the block verifiers compete in the second stage with each other to satisfy the strategy of the leader. This competitive interaction between followers is represented using a non-cooperative subgame. After the followers respond to the leader, each by its best strategy that is represented by the pair of verification price, denotes $\pi_j$, and verification delay, denoted $L_j$. Each verifier is considered rational and aims to optimize its own utility by maximizing the amount of incentive that it can earn in return for the block verification.

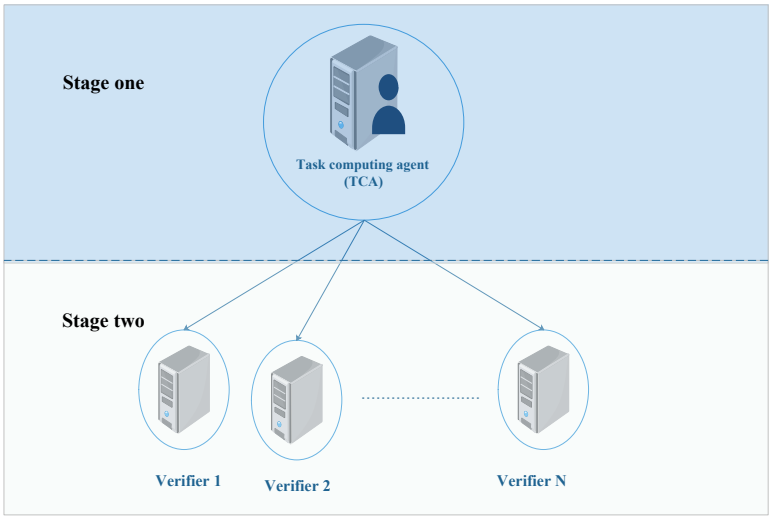


Figure 3.4: Two stages Stackelberg game for BC-based incentive caching .

The utility functions of the leader and followers are provided in the following two subsections. Utility is defined as a general function, such as [53]:

$$Utility = Reward - Cost. \tag{14}$$

### 3.5.1 TCA's utility model

In this part, we define the factors that might influence leader reward, cost, and the leader utility while following the previously described strategies. The utility function of the TCA may be described as the reward, denotes $R_{TCA}$, obtained at the ending of the block verification minus the cost, denoted $C_{TCA}$, that will be distributed to the followers.

#### 3.5.1.1 Reward

The leader reward (TCA) is primarily determined in function of its level of satisfaction when receiving the verifiers' responses. The TCA reward is defined as a monotonically increasing function of the number of verifiers and their reputation, plus the verification time or latency. $R_{TCA}$ is given by:

$$R_{TCA} = \sigma \times n \times \sum_{j=1}^{n} \left(\tau_j - L_j\right) \tag{15}$$

Where :

- $n$ is the number of the followers,

- $L_j$ is the verification response delay (verification latency) time,

- $\tau_j$ is the verification threshold, i.e., the maximum tolerated delay for block verification (i.e., the verification deadline),

- $\sigma$is the standard deviation of the followers' reputation score and is calculated as follows:

$$\sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\rho_j - \bar{\rho}\right)^2}. \tag{16}$$

- where $\rho_j$ is the reputation score of verifier $j$.

Initially, TCA sorts verifiers according to their reputation score using ZipF law and defines for each verifier $j$ the value of $\tau_j$ based on its reputation score and distance.

#### 3.5.1.2 Cost

The cost of the TCA is expressed as a monotonic increasing function of the sum of incentive rewards paid to followers as an incentive to encourage them to participate more efficiently in the block verification. The cost of the TCA ($C_{TCA}$) is given :

$$C_{TCA} = \log\left(\sum_{j=1}^{n}\log\left(1+\pi_j\left(\tau_j - L_j\right)\right)\times\sigma\right) \tag{17}$$

$\pi_j$ : is the gain of the follower $j$.

The utility function of the TCA (the leader), denoted $U_{TCA}$, may be described as an increasing function in terms of TCA gain and a decreasing function of TCA cost. $U_{TCA}$ is given by:

$$U_{TCA} = R_{TCA} - C_{TCA} \tag{18}$$

Where $R_{TCA}$and $C_{TCA}$ are described in 15 and 17, respectively.

### 3.5.2 Verifiers' utility model

The utility of verifiers $v_j$can be expressed as a general function in terms of the received reward, denoted $R_{vj}$, and the resource cost spent verifying the block, denoted $C_{vj}$.

#### 3.5.2.1 Reward

Each verifier seeks to maximize its reward by maximizing the earned incentive. The gain of the verifier reward, is stated as a monotonic increasing function in terms of earned incentive, verification time and the reputation score. The verifier's reward is designed to increase rapidly for low verification delay and high earned incentive, and increase slowly when delay is smaller than the block transmission time. $R_{vj}$ given by:

$$R_{vj} = \log\left(1 + \pi_j\left(\tau_j - L_j\right)\right) \times \sigma. \tag{19}$$

Where $L_j$ is calculated as follow :

$$L_j = T_i + \frac{Task_i}{\lambda_i} + l_{\text{auditors}} \tag{20}$$

- $T_i$: time of receiving the $Task_i$.
- $\frac{Task_i}{\lambda_i}$: The local verification time of this block is where i is the total number of CPU cycles necessary to complete the computation of a task.
- $l_{audit}$: time to receive all auditors ' responses.

#### 3.5.2.2 Cost

The main cost of the follower j, is equal to a part of earned incentives that share with validators in return for the block validation, plus the cost it paid to participate in the block verification, denote $C_{vj}$, (we design the verification paid, to discourage malicious verifiers to continuously playing the role of verifier), plus an energy parameter ($\gamma$). $\gamma$ represents the unity of energy consumed in a time slot to verify one unit of the block. $C_{vj}$ is given by:

$$C_{vj} = \log\left(1 + \frac{\gamma}{\left(\tau_j - L_j\right)} + CI_j\right) + \sum_{l=1}^{m} CA_l \tag{21}$$

$CA_l$ the amount that the verifier j must pay to the auditors in return of helping it in the block verification. Each verifier will reserve a percentage (equal to 1/3 of its reward) of its earning profits to motivate validators to validate the lock and sign with it the smart contract. $CA_l$ given by :

$$CA_l = \frac{\pi_j}{3 \times m}. \tag{22}$$

The utility function of the verifier $v_j$ , denoted $U_{vj}$,is expressed as an increasing function in terms of gain (reward) and a decreasing function in terms of the cost. $U_{vj}$ is given by:

$$U_{vj} = R_{vj} - C_{vj} \tag{23}$$

Where $R_{vj}$ and $C_{vj}$ are calculated using equations 19 and 21, respectively. The players must ensure that their utilities are always positive, with the cost always being less than the reward.

The next subsection details the non-cooperative subgame of followers.

### 3.5.3 Follower's non-cooperative Sub-game

We use a non-cooperative subgame to model the strategic competitive interaction between verifiers on the finite and limited announced incentive tokens of the leader $\mathfrak{R}$. Each verifier is considered rational and aims to optimize its utility. : The non-cooperative subgame between the followers is defined by $J = (V, S, U)$ such as :

- $V = \{v_1, ...., v_n\}$ represents the finite set of followers, i.e., the verifiers.
- $S = \{s_1, ...., s_n\}$ represents the finite set of verifiers' strategies where $s_j$ is the strategy of $v_j$, i.e., the verification delay and the corresponding verification cost.
- $U = \{u_1, ...., u_n\}$ denotes the finite set of follower's utility. $u_j$ is defined in equation 23.

Nash equilibrium (NE) is the most suitable solution for non-cooperative games. Nash equilibrium represents a mutually satisfactory situation, where no verifier would derivate and expect to get a better utility regarding the utility of their players [30]. For more details on NE, refer to Annex (A).In the next section, we will verify the existence of the Stackelberg equilibrium as well as compute the optimal strategies of the players.

## 3.6 Stackelberg equilibrium

The main purpose of our Stackelberg game is to derive the players' optimal strategies in the Stackelberg equilibrium (SE). Stackelberg equilibrium is defined as a scenario in which all players are satisfied. No player is motivated to diverge unilaterally to increase his utility [43]. In other words, neither leaders nor followers have an interest in deviating from the Stackelberg equilibrium point and expecting to enhance their utilities. In other words, neither the TCA has the interest to announce a lower verification deadline, nor the verifier aims to define a higher verification price while achieving a smaller verification delay.

For the solution of our sequential game, we apply backward induction like it is used in [54], [55]. This technique consists of starting by looking for the optimum reaction of the followers in the Nash equilibrium of a non-cooperative subgame before computing the best strategy of the leader.

### 3.6.1 Follower's optimal strategy

Considering the leader strategy, represented by the triplet ($\mathcal{R}$, $\tau_j$),the followers compete on the limited incentive of the leader and reply by their best strategies that satisfy the leader's utility. Indeed, each verifier with a high reputation score will try to verify the block as rapidly as possible and gather a large number of validators. As higher is the number of validated that

sign the smart contract with a verifier $j$, more will be the leader satisfaction and therefore the verifier income.

Each verifier aims to maximize its verification income by maximizing the verification cost, minimizing the verification latency and maximizing the number of validators that accept to sign the smart contract. Deriving the solution of the verifier subgame in NE, pass univetably by calculating the optimal verification cost ($\pi_j$) and the verification latency ($L_j$). The optimal price, $\pi^*{}_j$, is determined by solving the following optimization problem:

$$\begin{aligned} \pi_j^* = \arg\max_{\pi_j} U_{v_j}\left(\pi_j, L_j\right). \\ \text{s. } t \quad 0 < \pi_j < \mathcal{R} \\ L_j < \tau_j. \end{aligned} \tag{24}$$

$U_{vj}$ is defined in equation 23.

We follow the convex optimization method to investigate the optimal response of $v_j$ and therefore solve the follower optimization problem in 23. For that, we simply check if the followers' utility function is a concave optimization problem by computing the first and second order partial derivatives of the utility function $U_{vj}$ with respect to i for follower $i$.

- **Calculation of the first order derivative**

$$\frac{dU_{vj}}{d\pi_j} = \left(\left(\log\left(1+\pi_j\left(\tau_j - L_j\right)\right)\times\sigma\right) - \left(\frac{\gamma}{\tau_j - L_j} + CI_j + \frac{\pi_j}{3m}\right)\right)' \tag{25}$$

By applying differential formulas :

$$\frac{dU_{vj}}{d\pi_j} = \frac{d}{d\pi_j}\left(\left(\log\left(1+\pi_j\left(\tau_j - L_j\right)\right)\times\sigma\right) - \frac{d}{d\pi_j}\left(\frac{\gamma}{\tau_j - L_j} + CI_j + \frac{\pi_j}{3m}\right)\right) \tag{26}$$

We can deduce for j such that $i = \overrightarrow{1,m}$:

$$\begin{aligned} \frac{dU_{v1}}{d\pi_1} &= \left(\frac{\sigma\left(\tau_j - L_1\right)}{\pi_1\left(\tau_j - L_1\right)+1} - \frac{1}{3\times m} + 0 + \cdots + 0\right) \\ \frac{dU_{v2}}{d\pi_2} &= \left(0 + \frac{\sigma\left(\tau_j - L_2\right)}{\pi_2\left(\tau_j - L_2\right)+1} - \frac{1}{3\times m} + \cdots + 0\right) \\ \frac{dU_{vn}}{d\pi_n} &= \left(0 + 0 + \cdots + \frac{\sigma\left(\tau_j - L_n\right)}{\pi_n\left(\tau_j - L_n\right)+1} - \frac{1}{3\times m}\right) \end{aligned} \tag{27}$$

Finally, to obtain the general equation of the derivate, we simplify the equation as follows:

$$\mid \frac{dU_{vj}}{d\pi_j} = \left(\frac{\sigma\left(\tau_j - L_j\right)}{\pi_j\left(\tau_j - L_j\right)+1} - \frac{1}{3\times m}\right) \tag{28}$$

- **Calculation of the second order derivative**

$$\frac{d^2U_{vi}}{d\pi_i} = \left(\log\left(1+\pi_j\left(\tau_j - L_j\right)\times\sigma - \gamma_j + CI_j + \sum_{j=1}^{m}\frac{\pi_j}{3m}\right)\right)''$$
$$\frac{d^2U_{vj}}{d\pi_j} = \left(\frac{dU_{vj}}{d\pi_j}\right)' = \left(\frac{\sigma\left(\tau_j - L_j\right)}{\pi_j\left(\tau_j - L_j\right)+1} - \frac{1}{3\times m}\right)' \tag{29}$$
$$\frac{d^2U_{vi}}{d\pi_i} = -\frac{\sigma\left(\tau_j - L_i\right)^2}{\left(\pi_i\left(\tau_j - L_i\right)+1\right)^2}$$

The obtained value of the second derivative is negative $\left(-\frac{\sigma\left(\tau_j - L_j\right)^2}{\left(\pi_j\left(\tau_j - L_j\right)+1\right)^2}\right)$. thus by following the value of the sum remains < 0 ). This means that the utility function $U_{vj}$ is convenx relative to the verification charge price $\pi_j$ and the subgame admits at least one optimal solution that maximizes the verifier utility. Therefore, the optimal verification charge cost of a verifier can be reached using the Karush-Kuhn-Tucker constraint optimization condition (KKT condition). According to KKT conditions, we set the first derivative equal to 0 and solve the related equation. The optimal decision of the follower $j$ can be obtained as follow:

$$\frac{dU_{vj}}{d\pi_j} = \left(\frac{\sigma\left(\tau_j - L_j\right)}{\pi_j\left(\tau_j - L_j\right)+1} - \frac{1}{3\times m}\right) = 0$$
$$\frac{\sigma\left(\tau_j - L_j\right)}{\pi_j\left(\tau_j - L_j\right)+1} - \frac{1}{3\times m} = 0 \tag{30}$$
$$\frac{\sigma\left(\tau_j - L_j\right)}{\pi_j\left(\tau_j - L_j\right)+1} = \frac{1}{3\times m}$$

So:

$$\pi_i^* = \frac{3\times m\times\sigma\left(\tau_j - L_j\right)-1}{\left(\tau_j - L_j\right)} \tag{31}$$

is the best price for a verifier j that will be charged to be paid by the TCA for the verification of the block.

Now we must select the optimal verification price in NE by considering all of the follower's strategies. For this purpose, the reward received by a follower j is the total incentive ($\mathcal{R}$) announced by the TCA minus all the rewards earned by the remaining followers.

$$\pi_j = \mathcal{R} - \sum_{i\neq j}^{n}\pi_j \tag{32}$$

where:

$$\pi_1 = \mathcal{R} - (\pi_2 + \pi_3 + \ldots + \pi_n) \tag{33}$$

We can therefore write the following system of linear equations:

$$\begin{aligned}
0\times\pi_1+1\times\pi_2+1\times\pi_3+\ldots+1\times\pi_n &= \mathcal{R}-\frac{3\times m\times\sigma\left(\tau_1-L_1\right)-1}{\left(\tau_1-L_1\right)}\\
1\times\pi_1+0\times\pi_2+1\times\pi_3+\ldots+1\times\pi_n &= \mathcal{R}-\frac{3\times m\times\sigma\left(\tau_2-L_2\right)-1}{\left(\tau_2-L_2\right)}\\
&\vdots\\
1\times\pi_1+1\times\pi_2+1\times\pi_3+\ldots+0\times\pi_n &= \mathcal{R}-\frac{3\times m\times\sigma\left(\tau_n-L_n\right)-1}{\left(\tau_n-L_n\right)}
\end{aligned} \tag{34}$$

We transform the previous system of linear equations to a matrix equations system and we get:

$$\begin{bmatrix} 0 & 1 & 1 & \cdots & 1\\ \vdots & 0 & 1 & 1 & \vdots\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 1 & \cdots & 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \pi_0\\ \pi_1\\ \vdots\\ \vdots\\ \pi_n \end{bmatrix}=\begin{bmatrix} \mathcal{R}-\frac{3\delta\left(\tau_j-l_1\right)-m}{m\left(\tau_j-l_1\right)}\\ \mathcal{R}-\frac{3\delta\left(\tau_j-l_2\right)-m}{m\left(\tau_j-l_2\right)}\\ \vdots\\ \vdots\\ \mathcal{R}-\frac{3\delta(\tau_n-l_n)-m}{m(\tau_n-l_n)} \end{bmatrix}$$

The reward in the Nash equilibrium of each verifier is solved by applying the rule of Cramer's rule $\pi_m=\frac{det(Dm)}{det(D)}$ is the determinant of the matrix D and $det(Dm)$ is the determinant of the matrix $Dm$ which is formed by replacing the $Dm$ column of $D$ with the column vector $C$.

$$\det(D)=\begin{bmatrix} 0 & 1 & 1 & \cdots & 1\\ \vdots & 0 & 1 & 1 & \vdots\\ 1 & 1 & 0 & 1 & 1\\ 1 & 1 & 1 & 0 & 1\\ 1 & \cdots & 1 & 1 & 0 \end{bmatrix}=v-1 \tag{35}$$

We will calculate det(Dm)

$$\det\left(D_1\right)=\begin{bmatrix} \mathcal{R}-\frac{3\delta\left(\tau_j-l_1\right)-m}{m\left(\tau_j-l_1\right)}11\cdots1\\ \mathcal{R}-\frac{3\left(\tau_j-l_2\right)-m}{m\left(\tau_j-l_2\right)}01\cdots1\\ \vdots\\ \vdots\\ \mathcal{R}-\frac{3\delta(\tau_n-l_n)-m_1}{m(\tau_n-l_n)}11\cdots0 \end{bmatrix}$$

for $D1 : \det(Dm) = -\mathcal{R} - 8a + b + c + d + f + g + h + j + e + i$ where:

$$A = \begin{bmatrix} \mathcal{R}-a & 1 & \cdots & 1 & 1 \\ \mathcal{R}-b & 0 & \cdots & 1 & 1 \\ \mathcal{R}-c & 1 & \cdots & 1 & \vdots \\ \mathcal{R}-A & 1 & \cdots & 1 & \vdots \\ \mathcal{R}-e & 1 & \cdots & 1 & \vdots \\ \mathcal{R}-f & \vdots & \cdots & 1 & 1 \\ \mathcal{R}-g & 1 & \cdots & 1 & 1 \\ \mathcal{R}-h & i & \cdots & 1 & 1 \\ \mathcal{R}-i & 1 & \cdots & 0 & 1 \\ \mathcal{R}-j & 1 & \cdots & 1 & 0 \end{bmatrix}$$

for $D2 : \det(D2) = -\mathcal{R} + a - 8b + c + d + f + g + h + j + e + i$. where :

$$A = \begin{bmatrix} 0 & \mathcal{R}-a & \cdots & 1 & 1 & \\ 1 & \mathcal{R}-b & \cdots & 1 & 1 & \\ \vdots & \mathcal{R}-c & \cdots & 1 & \vdots & \\ \vdots & \mathcal{R}-A & \cdots & 1 & \vdots & \\ \vdots & \mathcal{R}-e & \cdots & 1 & \vdots & \\ \vdots & \mathcal{R}-f & \vdots & \cdots & 1 & 1 \\ \vdots & \mathcal{R}-g & \cdots & 1 & 1 & \\ \vdots & \mathcal{R}-h & \cdots & 1 & 1 & \\ \vdots & \mathcal{R}-i & \cdots & 0 & 1 & \\ \vdots & \mathcal{R}-j & \cdots & 1 & 0 & \end{bmatrix}$$

And so on for the other cases. for $D3 : \det(D3) = -\mathcal{R} + a + b - 8c + d + f + g + h + j + e + i$. for $D4 : \det(D4) = -\mathcal{R} + a + b + c - 8d + f + g + h + j + e + i$. for $D10 : \det(D10) = -\mathcal{R} + a + b + c + d + f + g + h + j + e - 8i$. We can deduce that [56] :

$$\det(A) = -\mathcal{R} + \beta_j$$

We recalculate the determinants of the matrices, and at the end, we can generalize the calcu-

lation of $\beta_j$ [57].

```
T[0] = a;
T[1] = b;
T[2] = c;
.....
for (j = 0;j < T[j];j + +)
for (i = 1;i < N)
  T[j] = T[j]*(−8);
end
end
T[j] = T[0] + T[1] + ... + T[N]
```

$$\pi_m^{NE} = \frac{\det(Dm)}{\det D} = \frac{-\mathcal{R} + \beta_j}{v-1}. \tag{36}$$

### 3.6.2 Leader optimal strategy

The TCA intends to minimize the verification time of its block while maximizing the number of highly trusted verifiers. As a result, after getting all of the best responses from the followers (verifiers). TCA will adjust the optimal verification time deadline (threshold $\left(\tau_j\right)$ ) for each verifier j by solving the following optimization problem:

$$\begin{gathered} \tau_j{}^* = \arg\min_{\pi_i} U_{TCA}(\tau). \\ \text{s.t } L_j < \tau_j. \\ \mathcal{R} \geq \sum_{j=1}^{n} R_{vj} \end{gathered} \tag{37}$$

We have :

$$U_{TCA} = \left( \sigma \times n \times \sum_{j=1}^{n} \left(\tau_j - L_j\right) - \sum_{j=1}^{n} \log\left(1 + \pi_j \left(\tau_j - L_j\right)\right) \times \sigma \right) \tag{38}$$

To solve the leader optimization problem in 38, we first replace the quantity $\pi_i$ by $\pi_m^*$ . So we obtain the following equation of the TCA utility:

$$U_{TCA} = \left( \sigma \times n \times \sum_{j=1}^{n} \left(\tau_j - L_j\right) - \sum_{j=1}^{n} \log\left(1 + \frac{-\mathcal{R} + \beta_j}{n-1} \left(\tau_j - L_j\right)\right) \times \sigma \right) \tag{39}$$

Next, we use convex optimization to prove the existence of the Stackelberg equilibrium and calculate the optimal strategy of the leader. Therefore, we start by calculating the first and second derivatives of the utility function $U_{TCA}$ and then derive the optimal value of $\tau_j$

- **Calculation of the first order derivative**

$$\frac{dU_{TCA}}{d\tau_j} = \left( \sigma \times n \times \sum_{j=1}^{n} \left( \tau_j - L_j \right) - \sum_{j=1}^{n} \log \left( 1 + \frac{-\mathcal{R} + \beta_j}{v-1} \left( \tau_j - L_j \right) \right) \times \sigma \right) \tag{40}$$

Application of differential formulas:

$$\frac{dU_{TCA}}{d\tau_j} = \left( -\frac{dU_{\tau CA}}{d\tau_j} \log \left( 1 + \frac{-\mathcal{R} + \beta_j}{v-1} \left( \tau_j - L_j \right) \right) \right) + \sigma \times n \left( \frac{dU_{\tau CA}}{d\tau_j} \left( -L_j \right) + \frac{dU_{\tau CA}}{d\tau_j} \left( \tau_j \right) \right) \tag{41}$$

We obtain:

$$\frac{dU_{TCA}}{d\tau_j} = \sigma \left( \frac{-\mathcal{R} + \beta_j}{\left( \tau_j - L_j \right) \times \left( -\mathcal{R} + \beta_j \right) - v + 1} + n \right) \tag{42}$$

- **Calculation of the second order derivative**

$$\frac{d^2 U_{TCA}}{d\tau_j} = \left( \sigma \times n \times \sum_{j=1}^{n} \left( \tau_j - L_j \right) - \sum_{j=1}^{n} \log \left( 1 + \frac{-\mathcal{R} + \beta_j}{v-1} \left( \tau_j - L_j \right) \right) \times \sigma \right)''$$

We obtain:

$$\frac{d^2 U_{TCA}}{d\tau_j} = \left( \frac{dU_{TCA}}{d\tau_j} \right)' = \sigma \left( \frac{-\mathcal{R} + \beta_j}{\left( \tau_j - L_j \right) \times \left( -\mathcal{R} + \beta_j \right) - v + 1} + n \right)$$

$$\frac{d^2 U_{TCA}}{d\tau_j} = \left( \frac{n \times \left( -\mathcal{R} + \beta_j \right)^2}{-v + \left( -\mathcal{R} + \beta_j \right) \left( \tau - L_j \right)^2 + 1} \right). \tag{43}$$

We found that the second derivative is positive. As a result, the function is concave and the SE exists. The KKT condition is used to achieve the optimum TCA choice. To do this, we use the first derivative $\frac{dU_{TCA}}{d\tau_j}$ and obtain:

$$\frac{dU_{TCA}}{d\tau_j} = \sigma \left( \frac{-\mathcal{R} + \beta_j}{\left( \tau_j - L_j \right) \times \left( -\mathcal{R} + \beta_j \right) - v + 1} + n \right) = 0.$$

$$\tau_j^* = \frac{-\beta_j \left( L_j \times n + 1 \right) + n \left( L_j \times \mathcal{R} + v - 1 \right) + r}{n \times \left( \mathcal{R} + \beta_j \right)}. \tag{44}$$

### 3.6.3 Pseudo Algorithm to find the Stackelberg equilibrium

**Algorithm :** Pseudo Algorithm to find the Stackelberg equilibrium

**Initialization**

- Let $n$ be the set of verifiers $V_i$ such that $i = 1..n$.
- Let $m$ be the set of auditors $A_j$ such that $j = 1..m$.
- Set the value of $R$ (Announced Incentive) to an initial value (i.e., 6000)
- Let $L_i$ be the latency of verifier $V_i$.
- Let $\tau_i$ be the adjusted treshold time for verifier $V_i$.
- Let $CA_i$ be the amount payed by verifier $V_i$.

**End Initialization**

**Begin**

1. **For** each $V_i$ **do**
   a) Calculate $L_i$ using equation (20)
   b) Leader Adjust $\tau_i$ using equation (44)
   c) Calculate $\pi_i^*$ using equation (31)
   d) Calculate $\pi_i^{NE}$ using equation (36)
2. **For** each $A_j$ **do**
   a) Calculate $CA_j$ using equation (22)
3. **End For**
4. **End For**

**End**

## 3.7 Conclusion

In this chapter, we proposed a new trust management system based on BlockChain technology to secure resource allocation. Our solution is divided into two main contributions, the first of which includes a blockchain-based trust management mechanism for task offloading in MEC. The second contribution for enhancing the block verification of the BC consensus mechanism. We have used a Stackelberg game with a non-cooperative subgame to model the strategic interaction between the block manager (TCA) and block verifiers. Convex optimization and backward induction are used to compute the players' strategies in Stackelberg equilibrium.

In the next chapter, we will present the implementation of our blockchain-based incentive mechanism for the proposed blockchain trust management for task offloading in mobile edge computing.

CHAPTER 4

# Implementation and Simulation Results

## 4.1 Introduction

In this chapter, we implement our incentive mechanism for the proposed blockchain-based trust management for task offloading in mobile edge computing. The main aim is to investigate the performance of our blockchain-based incentive mechanism under different parameters and through a variety of experiments.

## 4.2 Simulation tools and parameters

This section describes the various hardware and software tools used throughout the implementation of the proposed system. For our implementation, we chose Python for its excellent libraries that allow us to model, simulate, and implement our blockchain-based trust management for task offloading.

### 4.2.1 Simulation language: Python

Python is an open-source programming language created by programmer Guido van Rossum in 1991 [58]. The logo of python is illustrated in figure 4.1. It is"an interpreted, object-oriented, high-level programming language with dynamic semantics " [59]. Its popularity in the developer community comes from its simplicity of building applications in several domains that run on various platforms, as well as the availability of a very well-designed open-source library used for multiple sub-domains of computing.

Figure 4.1: Logo of Python.

As an IDE, we used Microsoft visual studio (VS Code) for its extension which makes it an excellent Python editor.

### 4.2.2 Simulation tool: visual studio code (VS Code)

Visual Studio Code is an open-source code editor developed by Microsoft that supports many languages through extensions such as C, C, C++, HTML, Java, JavaScript, Python, etc. It runs on Windows, Mac OS, and Linux. It provides developers with an integrated development environment with tools for advancing technical projects from editing to building and debugging [60]. The logo of visual studio code is illustrated in 4.2.

Figure 4.2: Logo of Visual Studio Code.

### 4.2.3 Simulation hardware configuration

We used a personal laptop with the specifications shown in table 4.1 to release our simulation.

| Characteristics | Processor | RAM | Hard Disk | Operating system |
|---|---|---|---|---|
| PC | Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz 2.71 GHz | 8 Go | 512 Go | Windows 10 X64 Bits |

Table 4.1: Configuration and setup of the used machine in simulation.

## 4.3 Simulation parameters

For the different experiments, we consider one TCA and 100 verifiers for our Stackelberg game-based approach. Each verifier has a set of 50 auditors (validators), a reputation score between 1 and 10, and a verification time between 5 and 20. Without loss of generality and for the sake of simulation, we assume that the verifier with a high reputation score has a short verification time. The cost paid to participate in the block verification is 2 and consumes 0.1 of resources. We consider that the TCA announces 60000 tokens as a total incentive to encourage and reward verifiers. Table 4.2 summarize the key simulation parameters:

| Parameter | Description | Value |
|---|---|---|
| n | Number of verifiers | 100 |
| m | Number of validators | 50 |
| $\rho$ | Reputation score | [1,10] |
| L | Verification response deadline | [5,20] |
| CI | Cost of participation in block verification | 2 |
| $\sigma$ | Consumed ressources | 0.1 |
| R | Total incentive announced by TCA | 60 000 Tokens |

Table 4.2: Simulation parameters.

## 4.4 Presentation of the application interface

Figure 4.3 shows our application's main interface. Through this window, the user can select the experiment with which he wishes to begin using the application, input the simulation settings, and then click the "Create charts" button to display the results.


Figure 4.3: The main application interface on python.

## 4.5 Simulation results

In this subsection, we will investigate the following experiments:

1. The impact of the number of verifiers on the utility function of the leader.
2. The impact of reputation variation on the utility function of the leader.
3. The impact of reputation variation on the utility function of the followers.
4. The impact of total incentive on the utility function of the leader.
5. The impact of the price on the utility function of verifiers.
6. The impact of the price on the utility function of the leader.
7. The Impact of the number of verifiers on the threshold and latency

### 4.5.1 Impact of the number of verifiers on the utility function of the leader (TCA )

In figure 4.4, we study the impact of varying the number of verifiers on the utility of the leader (TCA).

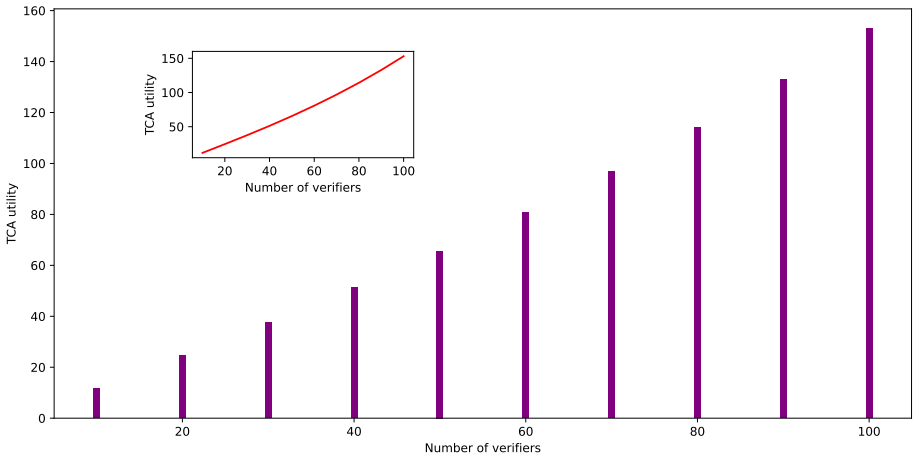


Figure 4.4: Impact of the number of verifiers on the utility of the leader (TCA).

The results obtained in figure 4.4 show that as the number of verifiers increases, the utility of the TCA increases. This is due to the fact that when the number of verifiers increases the gain of TCA as a direct result of the enhancement of the TCA satisfaction, the utility increases, see equation 15. More is the number of verifiers, the better will be the credibility of the block verification, and the higher will be the utility of TCA.

### 4.5.2 Impact of varying verifiers' reputations on the verifiers' utility

Figure 4.5 shows the study of the impact of reputation variation on the utility of followers. In this experiment, we increase each time the reputation of verifiers ([1,10], [10,20], and then [20,30]) and study the change in the utility average of verifiers.

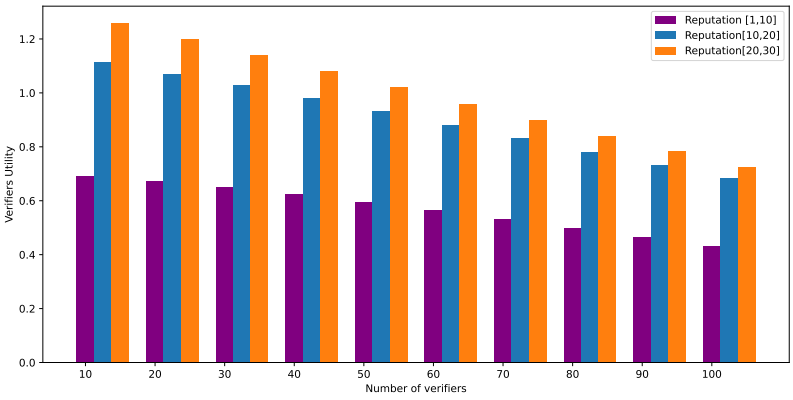


Figure 4.5: Impact of reputation variation on the utility of verifiers.

We observe from the results in figure 4.5 that the utility of the verifiers increases with the increase of reputation ($\rho_j$). This means that the standard deviation of the reputations increases and the verifiers with a high reputation will have more gain. Indeed, as reputation increases, the verifier will get more incentive income contributing to the improvement of his reward. knowing that verifiers are ranked according to their reputation in a descending manner which means the first verifiers are those who have a high reputation, which expresses the decrease of the verifier's utility.

### 4.5.3 Impact of varying verifiers' reputations on the TCA's utility

Figure 4.6 shows the study of the impact of reputation variation on the utility of the leader. In this experiment, we increase each time the reputation of verifiers ([1,10],[10,20], and then [20,30]) and study the impact on the TCA's average utility.

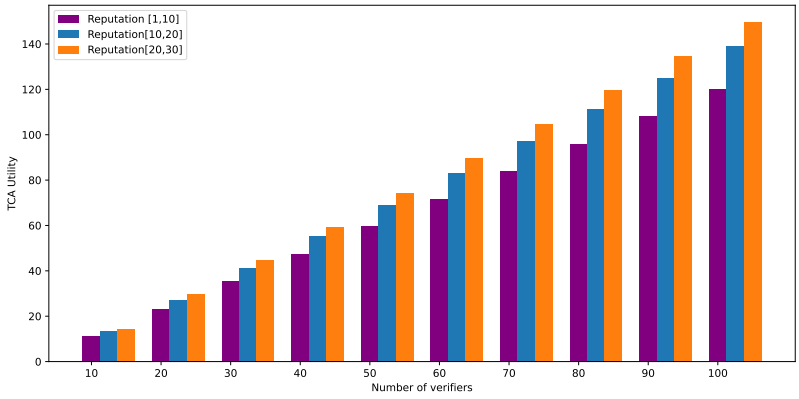


Figure 4.6: Impact of reputation variation on the utility function of the leader ( TCA ).

We observe from the results in figure 4.6 that the utility of the leader increases with the increase of the verifier's reputation ($\rho_j$). This means that the standard deviation of the reputations increases and the leader will have more gain and its reward will be improved. Indeed, as the number of verifiers increases, the utility of the TCA increases.

### 4.5.4 Impact of the total incentive on the verifiers' utility

Figure 4.7, shows the study of the impact of the total incentive announced by the TCA on the utility average of the verifiers for different numbers of verifiers.

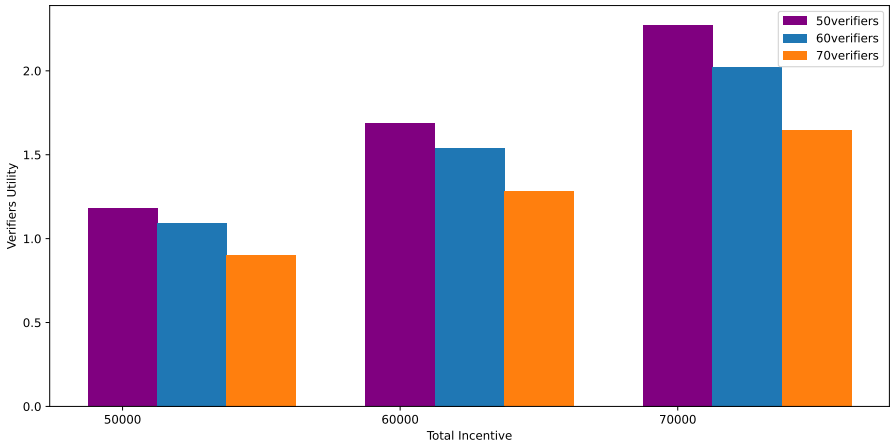


Figure 4.7: Impact of the total incentive on the utility function of verifiers.

The results in figure 4.7 clearly show that the verifier's utility increases with the increase of the total incentive given by the leader. This makes sense because the verifiers will have more incentive to share and therefore will generate more gain. Also, when the number of verifiers increases, the utility decreases since the total incentive will be shared among them due to the strategic competition between verifiers on this limited resource.

### 4.5.5 Impact of the price ($\pi_j$) on the verifiers' utility

In figure 4.8, we study the impact of changing the verification price ($\pi_j$) earned by the verifiers on their utilities.

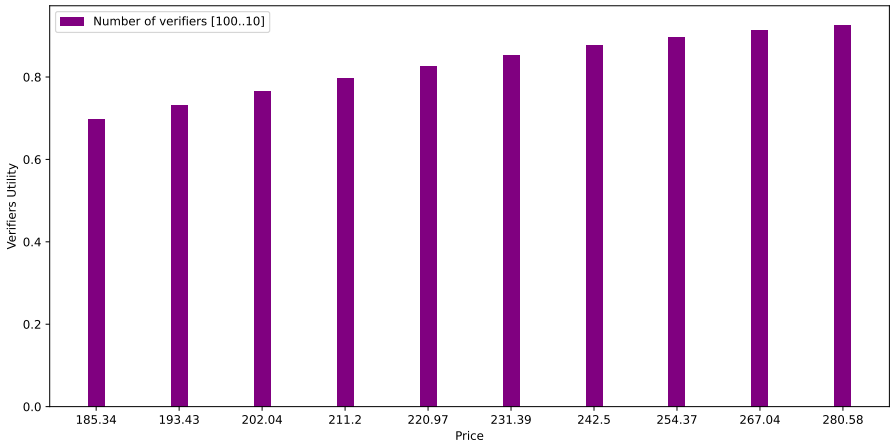


Figure 4.8: Impact of the price on the utility function of verifiers.

We observe from the results in figure 4.8 that as there are fewer verifiers, the greater is the gains earned because the verifiers will have more gains. Also, as the price increases, the utility average of verifiers increases. When the price rises, the reward rises as well, explaining the increase in the verifier's utility.

### 4.5.6 Impact of the price ($\pi_j$) earned by the verifiers on TCA's utility

In figure 4.9, we study the impact of the price ($\pi_j$) earned by the verifiers on the TCA's utility.

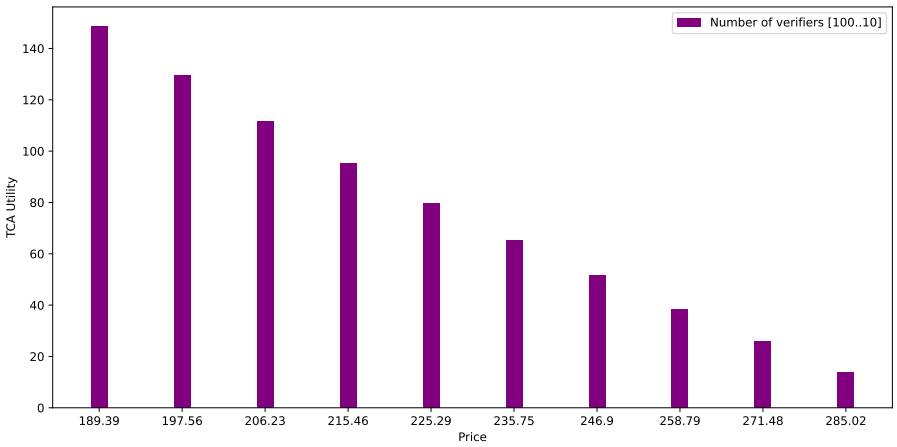


Figure 4.9: Impact of the price on the utility function of TCA.

We observe from the results in figure 4.9 that as there are less verifiers, the greater the gains earned. Also, as the price increases, the utility average of the TCA decreases. This can be explained by the following argument: when the number of verifiers increases the gain of TCA increases subsequently, the utility increases. see equation 15.

### 4.5.7 Impact of the number of verifiers on the threshold and latency

Figure 4.10 shows the study of the impact of the varying the number of verifiers on the verification threshold and the verification response delay (verification deadline).

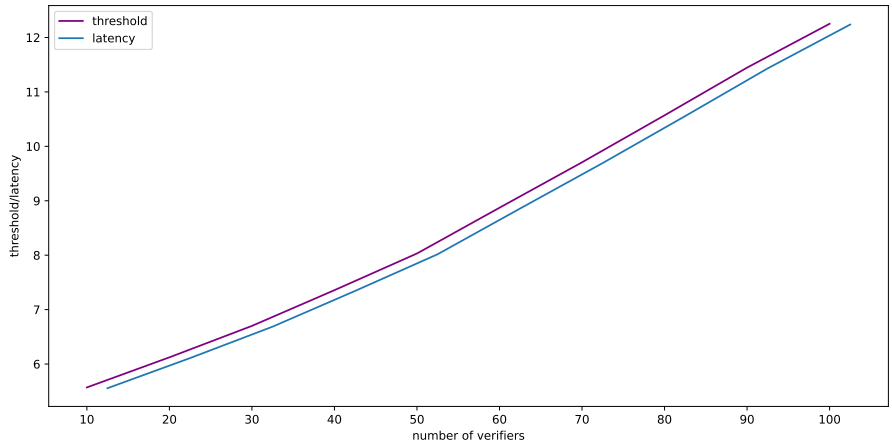


Figure 4.10: Impact of the varying number of verifiers on the threshold and latency.

We observe from the results in figure 4.10, that the optimal verification threshold is already adapted and successfully adjusted according to the verifier's verification response time. Therefore, our game-theoretic model allows the trust management system to perform an efficient incentive mechanism. In fact, the leader (TCA) can efficiently adjust the verification deadline according to the latency of each verifier, see equation 44. This allows for optimising the trust system parameters and optimising the leader utility.

## 4.6 Conclusion

This chapter is dedicated to the implementation and performance evaluation of our incentive mechanism for the proposed blockchain-based trust management for task offloading in mobile edge computing, as well as the interpretation obtained. The results show that system actors (TCA and verifiers) can reach optimal utilities and hence provide more efficient results.

# Conclusion

The work we've presented in this master report aims to solve the problem of trust management through a task offloading mechanism between the mobile devices and MEC. We proposed a blockchain-based trust management system to secure task offloading in mobile edge computing. Our system ensures the confidentiality of the user's identity as well as the reliability of the offloading computation. Also, through this incentive mechanism, the system encourages users to participate in blockchain consensus system.The Stackelberg game was then used to model the problem of inciting the verifiers and auditors to participate in the verification process. We used a convex optimization and backward induction to prove the Stackelberg equilibrium and derive the system optimal strategies.

We evaluated the performance of our incentive system under several experiments. The results obtained from the simulations showed that our model based on the Stackelberg game allows the TCAs and verifiers to reach optimal utilities and also allows for more efficient and reliable verification.

As future works, we intend to enhance our research and improve our approach to expanding our work on trust management by implementing a BC-trust management system and performing more advanced simulations to evaluate the performance of the incentive mechanisms using reinforcement learning. Also we hope extend our incentive mechanism into hierarchical block verification when TCA encourage the verifiers in one stage and the verifiers encourage the auditors in the sconde stage. The game well be modeled using the hierarchical stackelberg game with two subgames.

Appendix A

# Game Theory

## .1 Definition

Game theory, often known as decision theory, is the mathematical study of the strategic interactions of several rational agents. The following important terms are included in this definition [61]:

- **Interaction:** there are several agents (also called players, decision makers, etc.) who interact, the contentment (also called payment, gain, utility, etc.) of each one does not depend not only on him, but also partly on the others.
- **Strategic:** players have a choice between several options.
- **Rational:** a player does not play just any old way, he seeks to optimize his payment (his interest).

A game is defined through four basic concepts: players, strategies and utilities:

- **Players:** the player is a fundamental entity in a game that can represent a physical or logical individual as well as a group of humans or entities who participate in the game with a finite set of players represented by N. These entities must make decisions and perform actions in accordance with the game's rules in order to achieve a certain result. Players' interests are frequently varied and contradictory [62], [63].
- **Strategy (action/choice):** strategy represents one of a player's available actions or options [63]. It anticipates what the player should do in each situation that he will face. The strategy completely characterizes a player's conduct; it is based on an anticipatory process of selecting and performing actions to achieve a goal [62].

- **Utility:** It is a metric that represents a player's level of satisfaction.The utility must describe the agent's subjective degree of satisfaction and indicate the optimum scenario that maximizes a player's benefit [63]. This benefit or gain might be material, monetary, or logical.

- **Equilibrium:** Nash's equilibrium is the most commonly utilized solution in game theory. A Nash equilibrium (NE) is a condition in which no player has a motivation to modify his or her strategy unilaterally while taking into account the strategies of the other players [64]. A Nash equilibrium is a set of strategies $S = \{S_1, \ldots, S_n\}$ such that for any player $i$ and any strategy [65]:

$$S_i \in S, \text{ if } u_i(S_i, S_{i-1}) \geq u_i(s_i, S_{i-1})$$

## .2 Best response function

The best response function of player $i$ is the function $B_i$, which associates the strategies of player that maximize its utility with each combination of strategies of the other players $s_i$ [65]:The best response function of player $i$ is the function $B_i$, which associates the strategies of player that maximize its utility with each combination of strategies of the other players $S_{-i}$ [65]:

$$B_i(s_{-i}) = \{s_i \in S_i \text{ while } \mid u_i(s_i, s_{-i}) \geq u_i(s_i, s_{i-1}), \text{ for each } s_i \in S_i\}$$

## .3 Sequential Game

The sequential game is one of the game theory typologies, where the players take turns in sequential games. There are first and second players, and so on. At each move, each player is aware of all previous movements, including the latest move of the other players [66]. The participants choose their actions one by one. These games are not always totally fair in the sense that playing first generally provides a modest or even very minor advantage (but occasionally also a disadvantage) [66]. A sequential game is characterized by [66]:

- A set of players.

- A sequence, information, and actions made available to each player at the moment of play.

- The payoff at the end of the game, which depends on the history of the game.

### .3.1 Stackelberg Game

The Stackelberg game is a sequential game in which one player is called as the leader. The other players are called followers. In a Stackelberg game, the leader reveals a strategy first, and then the followers react logically to optimize the leader's actions [67].

- **Players type:** the Stackelberg game is designed to simulate two different sorts of competitive players. The leader is the game's starter by selecting an action from a set A1, and the second player, known as the follower, traces the leader's action and selects an action from a set A2 [68]

- **The Stackelberg equilibrium (SE):** is a mutually satisfied state (decision) from which neither the leader nor the followers choose to deviate unilaterally. The leader desires Stackelberg equilibrium [4], [6].

- **Sub-game:** a subgame is a component (a subset) of a game that has the criteria in game theory:

  - It has a single initial node that is the only member of that node's information set.
  - If a node is contained in the subset, all of its successors are also contained in the subset.
  - If a node in a particular information set is in the subset, then all members of that information set belong to the subgame.

- **Backward induction:** is a recurring reasoning method that is commonly employed in game theory to argue the presence of equilibrium in sequential games with complete knowledge. Indeed, it is necessary to begin searching for the equilibrium at the end (the last subgame) in order to arrive at the global equilibrium at the beginning (the global game).

# Bibliography


[1] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *Ieee Access*, 5:6757–6779, 2017.

[2] Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Blockchain for internet of things: A survey. *IEEE Internet of Things Journal*, 6(5):8076–8094, 2019.

[3] J Hurwitz, R Bloor, M Kaufman, and F Halper. Cloud computing for dummies,‖ wiley publishing. *Inc., Indianapolis, Indiana*, 2010.

[4] Sarfraz Nawaz Brohi and Mervat Adib Bamiah. Challenges and benefits for adopting the paradigm of cloud computing. *International Journal of Advanced Engineering Sciences and Technologies*, 8(2):286–290, 2011.

[5] Rajesh Kumar Goutam. Cloud computing characteristics.

[6] Guangshun Li, Jiping Wang, Junhua Wu, and Jianrong Song. Data processing delay optimization in mobile edge computing. *Wireless Communications and Mobile Computing*, 2018, 2018.

[7] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[8] Ejaz Ahmed and Mubashir Husain Rehmani. Mobile edge computing: opportunities, solutions, and challenges, 2017.

[9] Inés Sittón-Candanedo, Ricardo S Alonso, Óscar García, Lilia Muñoz, and Sara Rodríguez-González. Edge computing, iot and social computing in smart energy scenarios. *Sensors*, 19(15):3353, 2019.

[10] Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. In *2012 IEEE international conference on pervasive computing and communications*, pages 122–127. IEEE, 2012.

[11] Arslan Rasheed, Peter Han Joo Chong, Ivan Wang-Hei Ho, Xue Jun Li, and William Liu. An overview of mobile edge computing: Architecture, technology and direction. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(10):4849–4864, 2019.

[12] Arslan Rasheed, Peter Han Joo Chong, Ivan Wang-Hei Ho, Xue Jun Li, and William Liu. An overview of mobile edge computing: Architecture, technology and direction. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(10):4849–4864, 2019.

[13] Congfeng Jiang, Xiaolan Cheng, Honghao Gao, Xin Zhou, and Jian Wan. Toward computation offloading in edge computing: A survey. *IEEE Access*, 7:131543–131558, 2019.

[14] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *Mobile networks and Applications*, 18(1):129–140, 2013.

[15] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358, 2017.

[16] Qiang Tang, Haimei Lyu, Guangjie Han, Jin Wang, and Kezhi Wang. Partial offloading strategy for mobile edge computing considering mixed overhead of time and energy. *Neural Computing and Applications*, 32(19):15383–15397, 2020.

[17] Zhenjiang Zhang, Chen Li, ShengLung Peng, and Xintong Pei. A new task offloading algorithm in edge computing. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–21, 2021.

[18] Jin Wang, Jia Hu, Geyong Min, Wenhan Zhan, Qiang Ni, and Nektarios Georgalas. Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning. *IEEE Communications Magazine*, 57(5):64–69, 2019.

[19] Xue Lin, Yanzhi Wang, Qing Xie, and Massoud Pedram. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. *IEEE Transactions on Services Computing*, 8(2):175–186, 2014.

[20] Vincenzo De Maio and Ivona Brandic. First hop mobile offloading of dag computations. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 83–92. IEEE, 2018.

[21] Yuqing Li, Xiong Wang, Xiaoying Gan, Haiming Jin, Luoyi Fu, and Xinbing Wang. Learning-aided computation offloading for trusted collaborative mobile edge computing. *IEEE Transactions on Mobile Computing*, 19(12):2833–2849, 2019.

[22] Wenping Kong, Xiaoyong Li, Liyang Hou, and Yanrong Li. An efficient and credible multi-source trust fusion mechanism based on time decay for edge computing. *Electronics*, 9(3):502, 2020.

[23] Asma Lahbib, Khalifa Toumi, Anis Laouiti, Alexandre Laube, and Steven Martin. Blockchain based trust management mechanism for iot. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2019.

[24] Merrihan BM Mansour, Tamer Abdelkader, Mohamed Hashem, and El-Sayed M El-Horbaty. An integrated three-tier trust management framework in mobile edge computing using fuzzy logic. *PeerJ Computer Science*, 7:e700, 2021.

[25] Henk CA Van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.

[26] Tyrone Grandison and Morris Sloman. Specifying and analysing trust for internet applications. In *Towards the Knowledge Society*, pages 145–157. Springer, 2003.

[27] Cristiano Castelfranchi. Trust mediation in knowledge management and sharing. In *International Conference on Trust Management*, pages 304–318. Springer, 2004.

[28] Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *International Conference on Trust Management*, pages 77–92. Springer, 2005.

[29] Wenjuan Li, Jiyi Wu, Qifei Zhang, Keyong Hu, and Jing Li. Trust-driven and qos demand clustering analysis based cloud workflow scheduling strategies. *Cluster computing*, 17(3):1013–1030, 2014.

[30] Merrihan BM Mansour, Tamer Abdelkader, Mohamed Hashem, and El-Sayed M El-Horbaty. An integrated three-tier trust management framework in mobile edge computing using fuzzy logic. *PeerJ Computer Science*, 7:e700, 2021.

[31] Khushboo Tripathi, Neha Bhateja, and Ashish Dhillon. Changing the conventional banking system through blockchain. *The Smart Cyber Ecosystem for Sustainable Development*, pages 379–403, 2021.

[32] Laurent Leloup. *Blockchain: La revolution de la confiance*. Editions Eyrolles, 2017.

[33] Souad HABBANI and Imane EL FALLAHI. La blockchain pour la gestion des informations au sein des administrations publiques marocaines: Opportunités et défis. *International Journal of Accounting, Finance, Auditing, Management and Economics*, 3(1-2):412–425, 2022.

[34] Arshdeep Bahga and Vijay K Madisetti. Blockchain platform for industrial internet of things. *Journal of Software Engineering and Applications*, 9(10):533–546, 2016.

[35] Vishwani Patel, Fenil Khatiwala, Kaushal Shah, and Yashi Choksi. A review on blockchain technology: Components, issues and challenges. *ICDSMLA 2019*, pages 1257–1262, 2020.

[36] Sophie Drame-Maigne. *Blockchain and access control : towards a more secure Internet of Things*. Theses, Université Paris Saclay (COmUE), November 2019.

[37] Vivek Acharya, Anand Eswararao Yerrapati, and Nimesh Prakash. Oracle blockchain services quick start guide. *Packt, Mumbai*, 2019.

[38] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. Ieee, 2017.

[39] Haijun Liao, Yansong Mu, Zhenyu Zhou, Meng Sun, Zhao Wang, and Chao Pan. Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4051–4063, 2020.

[40] Dinh C Nguyen, Pubudu N Pathirana, Ming Ding, and Aruna Seneviratne. Secure computation offloading in blockchain based iot networks with deep reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 8(4):3192–3208, 2021.

[41] Xiaoyu Zhu, Yi Li, Li Fang, and Ping Chen. An improved proof-of-trust consensus algorithm for credible crowdsourcing blockchain services. *IEEE Access*, 8:102177–102187, 2020.

[42] Fábio Pereira, Paul Crocker, and Valderi RQ Leithardt. Padres: Tool for privacy, data regulation and security. *SoftwareX*, 17:100895, 2022.

[43] Jun Li, Wen Chen, Ming Xiao, Feng Shu, and Xuan Liu. Efficient video pricing and caching in heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 65(10):8744–8751, 2015.

[44] Jiawen Kang, Zehui Xiong, Dusit Niyato, Ping Wang, Dongdong Ye, and Dong In Kim. Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks. *IEEE Wireless Communications Letters*, 8(1):157–160, 2018.

[45] Shihong Zou, Jinwen Xi, Siyuan Wang, Yueming Lu, and Guosheng Xu. Reportcoin: A novel blockchain-based incentive anonymous reporting system. *IEEE access*, 7:65544–65559, 2019.

[46] Xiaoyu Zhu, Yi Li, Li Fang, and Ping Chen. An improved proof-of-trust consensus algorithm for credible crowdsourcing blockchain services. *IEEE Access*, 8:102177–102187, 2020.

[47] Audun Josang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In G Dobbie and V Estivill-Castro, editors, *Conference Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*, pages 85–94. Australian Computer Society, CD Rom, 2006.

[48] Jiawen Kang, Zehui Xiong, Dusit Niyato, Dongdong Ye, Dong In Kim, and Jun Zhao. Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory. *IEEE Transactions on Vehicular Technology*, 68(3):2906–2920, 2019.

[49] Thuat Do, Thao Nguyen, and Hung Pham. Delegated proof of reputation: A novel blockchain consensus. In *Proceedings of the 2019 International Electronics Communication Conference*, pages 90–98, 2019.

[50] Peter Gaži, Aggelos Kiayias, and Dionysis Zindros. Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 139–156. IEEE, 2019.

[51] Jiawen Kang, Zehui Xiong, Dusit Niyato, and Dong In Kim. Incentivizing secure block verification by contract theory in blockchain-enabled vehicular networks. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[52] Lun Li, Jiqiang Liu, Lichen Cheng, Shuo Qiu, Wei Wang, Xiangliang Zhang, and Zonghua Zhang. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2204–2220, 2018.

[53] Ahmed Alioua, Samiha Simoud, Sihem Bourema, Manel Khelifi, and Sidi-Mohammed Senouci. A stackelberg game approach for incentive v2v caching in software-defined 5g-enabled vanet. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[54] Jun Li, Wen Chen, Ming Xiao, Feng Shu, and Xuan Liu. Efficient video pricing and caching in heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 65(10):8744–8751, 2015.

[55] Fei Shen, Kenza Hamidouche, Ejder Bastug, and Merouane Debbah. A stackelberg game for incentive proactive caching mechanisms in wireless networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[56] Simoud Samiha and Bourema Sihem. *Un jeu de Stackelberg pour le problème de l'incitation à la mise en cache dans les réseaux d'internet de véhicules définis par logiciel.* PhD thesis, 2018.

[57] Ahmed Alioua, Roumayssa Hamiroune, Oumayma Amiri, Manel Khelifi, Sidi-mohammed Senouci, Mikael Gidlund, and Sarder Fakhrul Abedin. Incentive mechanism for competitive edge caching in 5g-enabled internet of things. *Computer Networks*, page 109096, 2022.

[58] Python : tout savoir sur le principal langage big data et machine learning. `https://www.lebigdata.fr/python-langage-definition`, 2022. Accessed: 2022-06-27.

[59] What is python? executive summary. `https://www.lebigdata.fr/python-langage-definition`. Accessed: 2022-06-27.

[60] Découvrir visual studio code. `https://www.blogdumoderateur.com/tools/visual-studio-code`. Accessed: 2022-06-27.

[61] Jianfeng Lu, Zhao Zhang, Jiangtao Wang, Ruixuan Li, and Shaohua Wan. A green stackelberg-game incentive mechanism for multi-service exchange in mobile crowdsensing. *ACM Transactions on Internet Technology (TOIT)*, 22(2):1–29, 2021.

[62] Nicole Berline, Alain Plagne, and Claude Sabbah. *Théorie des jeux: Introduction à la théorie des jeux répétés*. Editions Ecole Polytechnique, 2007.

[63] Mohamed S Abdalzaher, Karim Seddik, Maha Elsabrouty, Osamu Muta, Hiroshi Furukawa, and Adel Abdel-Rahman. Game theory meets wireless sensor networks security requirements and threats mitigation: A survey. *Sensors*, 16(7):1003, 2016.

[64] Ivan Pastine and Tuvana Pastine. L'équilibre de nash. In *La théorie des jeux en images*, pages 30–32. EDP Sciences, 2021.

[65] S ébastien Konieczny. Introduction à la théorie des jeux.

[66] Quan Wen. A folk theorem for repeated sequential games. *The Review of Economic Studies*, 69(2):493–512, 2002.

[67] Mohamed S Abdalzaher, Karim Seddik, Maha Elsabrouty, Osamu Muta, Hiroshi Furukawa, and Adel Abdel-Rahman. Game theory meets wireless sensor networks security requirements and threats mitigation: A survey. *Sensors*, 16(7):1003, 2016.

[68] Ming Hu and Masao Fukushima. Multi-leader-follower games: models, methods and applications. *Journal of the Operations Research Society of Japan*, 58(1):1–23, 2015.

[69] Daniel Bastos. Cloud for iot—a survey of technologies and security features of public cloud iot solutions. In *Living in the Internet of Things (IoT 2019)*, pages 1–6. IET, 2019.

[70] Rajeev Kumar Bedi, Jaswinder Singh, and Sunil Kumar Gupta. Current trends in cloud storage for resource constrained mobile devices. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1144–1149. IEEE, 2016.

[71] Haijun Liao, Yansong Mu, Zhenyu Zhou, Meng Sun, Zhao Wang, and Chao Pan. Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4051–4063, 2020.

[72] Fei Shen, Kenza Hamidouche, Ejder Bastug, and Mérouane Debbah. A stackelberg game for incentive proactive caching mechanisms in wireless networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[73] G Th Guilbaud. La theorie des jeux. *Revue d'économie politique*, 65(2):153–188, 1955.