Blockchain & Multi-Agent System: A New Promising Approach for Cloud Data Integrity Auditing with Deduplication

Mohamed El Ghazouani, Moulay Ahmed El Kiram and Latifa Er-Rajy

Computer Science Department, Laboratory LISI, Cadi Ayyad University, Morocco

***Abstract*:** Recently, data storage represents one of the most important services in Cloud Computing. The cloud provider should ensure two major requirements which are data integrity and storage efficiency. Blockchain data structure and the efficient data deduplication represent possible solutions to address these exigencies. Several approaches have been proposed, some of them implement deduplication in Cloud server side, which involves a lot of computation to eliminate the redundant data and it becomes more and more complex. Therefore, this paper proposed an efficient, reliable and secure approach, in which the authors propose a Multi-Agent System in order to manipulate deduplication technique that permits to reduce data volumes thereby reduce storage overhead. On the other side, the loss of physical control over data introduces security challenges such as data loss, data tampering and data modification. To solve similar problems, the authors also propose Blockchain as a database for storing metadata of client files. This database serves as logging database that ensures data integrity auditing function.

***Keywords*:** Blockchain, Integrity, Public Auditing, Deduplication, Multi-Agent Systems.

1. Introduction

Over the past few years, Cloud Computing provided a number of opportunities such as enabling services to be used without any understanding of their infrastructure, also data and services are managed remotely but they are accessible anywhere. The remote storage way can be considered as one of the most important Cloud services because it allows Cloud users to store their data from local storage systems to the remote Cloud while enjoying unstinted storage services.

Based on NIST's classification, private Cloud, community Cloud, public Cloud, and hybrid Cloud are the four major patterns of Cloud deployment [1][2] (figure 1):

- **Private Cloud**: The system is operated on behalf of one entity that can be an individual, company or organization.
- **Public Cloud**: Services are made available to the general public through the Internet. The resource offered in this type might be cost free or the services are rented for the customers according to their utilization.
- **Community Cloud**: Cloud infrastructure is controlled and used by people who share similar computing business or common interests such as missions, security requirements, policies or compliance requirements.
- **Hybrid Cloud**: It consists of the combination of public and private cloud services.

Since cloud computing can fulfill any IT need conceivable in a virtual way, the three cloud service models: SaaS, PaaS and IaaS, are necessary to indicate the role that a particular cloud service accomplishes, and how that service fulfills its role (Figure1).

- **SaaS** (Software as a Service) is a model of software deployment where the software/applications are provided to the customers as a service through a program interface or a web browser. The Cloud's client does not need to install IT infrastructure such as network, servers, operating systems and application software inside his company because they are hosted and managed in supplier's site.
- **PaaS** (Platform as a Service) delivers a computing platform where the client can create, execute, deploy and manage their applications.
- **IaaS** (Infrastructure as a Service) delivers computer infrastructure, typically a platform virtualization environment as a service. Rather than purchasing servers, software, storage, memory, processor or network equipment, clients buy those resources as fully outsourced services.

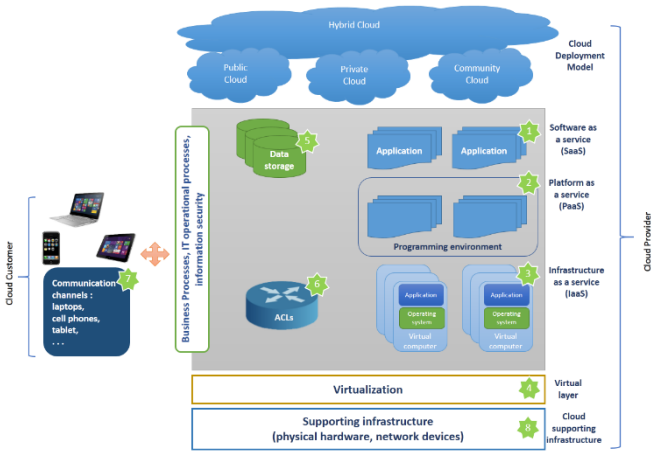

Figure 1. General architecture of Cloud Computing

Many cloud users upload and store replicated data. To solve such problem, cloud service providers try to maximize bandwidth and minimize storage space by applying data deduplication technique. A good management of data deduplication can be thought of as the best choice to ensure data storage efficiency. Data deduplication (called also intelligent compression or single-instance storage) is a technique that permits to eliminate redundant data and keep just one copy of each duplicated data before its transfer to the Cloud storage server (deduplication in the client side called source-based deduplication) or after it is transferred (deduplication in the server side called also target-based deduplication) which allows to reduce data volumes thereby reduces storage overhead.

On other side, once the data are stored in the Cloud storage servers, the data owners lose control over their data. Despite all tremendous benefits of cloud storage, cloud providers suffer from some security challenges, especially those related to the protection weakness of data integrity. This latter is considered as one of the most critical elements in any system. In fact, the service providers have to provide efficient audit proof to ensure data integrity in order to establish a solid confidence with clients. Thus, users should enable auditing methods to check the integrity of their outsourced data. Auditing is a process of analysis and verification, performed by an internal or external auditor, with the aim of presenting security vulnerabilities of a system. In this paper, the authors use the auditing process to check data integrity of the outsourced data.

In this paper, the authors propose a new approach that ensures efficient storage based on data deduplication and preserves data integrity auditing using Blockchain technology in a Cloud Computing environment. The structure of this paper is followed. Section 2 outlines the problem statement, presenting system model, threat model and model goals. In Section 3, various related works are discussed. After that, in section 4, different concepts used in our proposed method are presented. The section 5 provides a detailed description of our proposed approach. Section 6 includes the discussion and analysis. Finally, the conclusion and ongoing works are presented in section 7.

2. Problem Statement

In this section, the authors first describe the basic system model for cloud data storage and auditing. Then they will highlight the threat model. Finally, the authors fix the approach goals.

2.1 System Model

The basic system model consists of three entities: the data owner, the cloud service provider (CSP) and the third party auditor (TPA). The general architecture of the system model is shown in the figure 2.

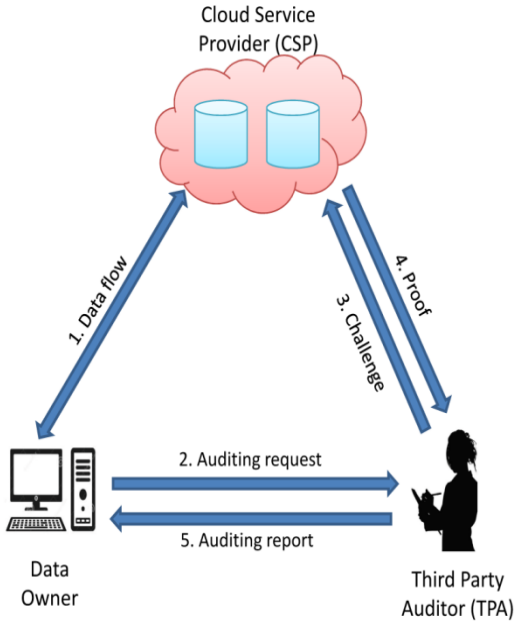

Figure 2. Architecture of system model

- **Data owner**: is the proprietor of data, it is the entity that uploads and stores the data on the remote cloud in order to reduce the burden of managing the data locally by itself. Data owner can sometimes store duplicated data.
- **Cloud Service Provider (CSP)**: is the entity that provides space to store client's data. Besides, CSP generates proofs of possession of data to the TPA.
- **Third party auditor (TPA)**: has the capabilities to access services afforded by CSP. Although, the data owner requests him to check the integrity of its data. Therefore, TPA interrogates the CSP for a challenge request to check the possession of data. Finally, he prepares a data possession report and sends it to the data owner as an auditing report.

2.2 Threat Models

Data owner considers TPA as a reliable and honest entity that will verify the integrity of his data. Thus, it is probably that TPA might be curious about these data. In this case, the TPA could be considered as a threat for data owner. Therefore, to assure the storage correctness of the owner's data at the cloud server, it will be necessary to have a protection mechanism, which ensures that TPA cannot learn anything about the owner's data.

On the other hand, CSP is not fully trusted. The latter can pose some threats to the owner's data. For example, in order to save space, CSP may remove data which have never been accessed without any notification to the data owner. Besides, the outsourced data may be tampered or even re-outsourced without notice by malicious CSP. CSP can also apply some changes on the owner's data wrongly owing to system failure, management errors or any other reason. However, it hides these mistakes to protect its image. While data are stored on the cloud servers and to respond to TPA's query, CSP can use an authentic pair of data block as a substitute to the queried data blocks just to pass out the audit. CSP can also get back the previous stored results of the data challenged just to generate the proof of possession of data and not to really query the owner's data.

2.3 Approach Goals

Motivated by data integrity and deduplication, the authors propose a new method for data storage and auditing in Cloud based on Multi-Agent system. The authors aim to achieve the following goals in the proposed method:

(1) **Server-side deduplication (Storage efficiency)**: eliminates duplicated data and it is performed via multi-agent system in the cloud server side, which reduces data volumes thereby reduces storage overhead.
(2) **Confidentiality**: ensures the confidentiality of owner's data against TPA during the auditing process.
(3) **Public auditing**: allows the TPA to check the correctness of the stored data in the Cloud.
(4) **Batch Auditing**: ensures that TPA performs multiple auditing tasks, in a simultaneous way, received from different users
(5) **Data Integrity**: ensures that the CSP cannot cheat and pass the auditing process without having stored the data intact.
(6) **Lightweight**: provides the model with low communication and computational overheads.

3. Related Work

For the past few years, a number of researchers have paid considerable attention to the problems of data deduplication and data integrity, and a series of schemes and approaches have been proposed in these fields. Ateniese et al.[3] provided a provable data possession (PDP) scheme. It is a model that allows the user to verify the correctness of the outsourced data without retrieving it. In this model, the server does not need full access to the entire file to generate the proof. Ateniese et al. [4] extended this protocol to E-PDP, which is 185 times faster than PDP. Another variation of PDP proposed by Juels and Kaliski [5] is a Proof of Retrievability (POR), where a proof is produced and a user can retrieve the file from the remote storage. This model suffers from computation overhead. The main drawback of the above protocols is that they do not allow dynamic data auditing. Hovavshacham and Brent Waters [6] proposed a compact proof of retrievability, which is a variant of PoR model. The proposed scheme supports stateless verification, challenge response protocol and public auditability. Erway et al.[7] proposed an efficient scheme for dynamic provable data possession (DPDP). It is the first scheme that supports dynamic auditing system. However, the main drawback of this scheme is that it cannot support public auditing. Wang et al.[8] resolved the above two problems by presenting a public and dynamic auditing scheme that is based on Merkle Hash Tree (MHT). It is a structured tree where the leaves are hashes of authentic data blocks. This protocol supports batch auditing. Howbeit, this scheme involves more computational costs during updating and auditing phases. Liu [9] had expanded MHT to rank-based MHT (R-MHT) with efficient verifiable fine-grained updates. Zhang[10] improved the MHT scheme to have a balanced update tree. Zhu et al.[11]presented an auditing scheme known as index-hash table based public auditing (IHT-PA). The main goal of this protocol is to minimize the computation and communication costs. However, it is inefficient for dynamic updating operations. Tian et al.[12] proposed a scheme based on dynamic hash table (DHT) which supports public and dynamic auditing. This protocol achieves better performance in the updating phases. Tang and Zhang [13] introduced a model that supports both private and public verifiability to check the integrity of user data stored on cloud server without downloading it. This model is known as verifiable data possession (PVDP). Aiping Li et al.[14] proposed an efficient method for achieving provable data integrity in cloud computing. This scheme supports dynamic operations and batch auditing. Yu et al. [15] proposed an identity-based auditing scheme for checking the integrity of outsourced data. However, Xu Z, Wu L, Khan MK, and alrevealed that this scheme is vulnerable to data recovery attack. Therefore, they proposed a secure and efficient identity-based public auditing scheme using RSA algorithm for cloud storage[16]. Lee KM et al. [17] presented a new data integrity checking scheme for remotely acquired and stored stream data.

Zheng and Xu [18] proposed a model that allows to remove the extra copies of the same file in PDP scheme. Li et al. [19] proposed a secure deduplication storage system that can support keyword search. Miao et al.[20] presented a secure multi-server-aided data deduplication protocol in cloud computing. These schemes mainly considered data deduplication, however, they did not mention data storage security. Yuan and Yu[21] proposed a public and constant cost public cloud storage auditing with deduplication (PCAD). This scheme supports batch auditing. Although this scheme handles data security storage and deduplication, but it does not consider the data block update.

4. Concepts Used in our Proposed Model

In the proposed method, the authors introduce the use of four concepts; Blockchain that ensures data integrity in Cloud Computing, Merkle Hash Tree that is a binary tree representing data structure used in Blockchain technology and Multi-Agent System that manages the complexity of deduplication process.

4.1 Blockchain

Blockchain is a quite recent technology that appeared in 2008 with the digital currency Bitcoin [22]. Blockchain refers to a chain of blocks that are digital containers on which are stored information of all kinds: transactions, contracts, titles of property ...etc. This technology allows storage and transmission of information, in a transparent and secure mode, without the participation of any central controlling station. Blockchain is a disintermediation tool that has the effect of reducing costs and streamlining exchanges. The storage is made in several servers, which implies the difficulty of being attacked.

Blockchain is characterized by a cryptographic protocol that allows all users to agree on the status of the registry, which guarantees its security. One of the most important advantages is that the Blockchain was invented to exchange value in a peer-to-peer fashion without an intermediary.

Blockchain is a database that stores transactions shared by all participated users of the system where transaction data is recorded permanently (Figure 3). A set of steps are performed before the creation of a new block in the Blockchain [23].

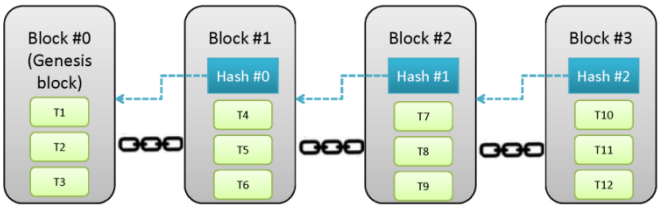

Figure 3. Blockchain structure

Blockchain consists of some generated blocks that are organized into a linked-list. Each block is linked to the previous one via a digest of the previous block called parent block. The first block, numbered as #0, of this linked-list is called the genesis block. Each new transaction is persistently being processed by distinguished nodes of the network, i.e. miners, according to techniques that depend on the type of Blockchain. Once the transaction is added to the end of the chain, no one can modify or erase the block. In the Blockchain of the Bitcoins this technique is called the "Proof-of-Work", for this reason it is considered to be tamper-proofed.

The Blockchain is able to effectively ensure integrity and authenticity of the exchanged data and especially auditability by providing a private layer where Cloud data are treated and stored in a reduced time. Therefore, confidence is established,

between the client and CSP, through solid cryptographic mechanisms and not via intermediaries. Blockchain can be considered as a tool for archiving and verifying data in a secure and lightweight manner.

Cloud Storage consists of a set of virtual machines. This can be successfully applied for Blockchain data structure. In our model, the Blockchain is considered as a database that stores all information for a file in form of blocks. This database serves as a logging database that ensures data integrity auditing function.

4.2 Merkle Hash Tree

Merkle Hash Tree comes from its inventor Ralph Merkle, who published the algorithm in 1979[24]. It is a specific binary tree that Bitcoin programmers use every day. Merkle Tree can be used to authenticate digital data with a lower computation and communication overhead. The main goal of the Merkle hash tree is to decompose the input data into a set of blocks of the same size. In Bitcoins, Merkle hash tree corresponds to a tree by block encompassing the whole transactions that are linked to the block. In the proposed method, the authors replace transactions by file-blocks. Each file gives rise to a Merkle hash tree. Thus, Merkle hash tree allows a digest of all the file-blocks linked to that block.

To understand more clearly Merkle hash tree, the authors gave the following example. Let us consider a file with 4 file-blocks fb1, fb2, fb3 and fb4. These initial leaves are called the leaves of Merkle's tree. They are processed in pairs recursively. The hashes values are computed as: H1 = h(fb1), H2 = h(fb2), H3 = h(fb3), H4 = h(fb4), H5 is the concatenation of H1 and H2, H6 is the concatenation of H3 and H4. Finally, to construct the parent node H7, that is at the top, the two nodes H5 and H6 are concatenated, this node is known as Merkle Root, it always summarizes them into 256 bits whatever the number of the file-blocks.

The authors can use the Merkle hash tree to authenticate any subset of file-blocks via all the sibling nodes on the path from the specific leaf node to the root.

As shown in the figure 4, file-blocks are not stored in the Merkle Tree, rather it's their hashes that are stored in each node. If a small bit is changed in any file-block, it will be certainly a significant difference in the resulting Merkle root.

In our method, each Merkle Tree, generated from the file-blocks corresponding to a file, is stored in a block in the Blockchain. The Merkle Root is fundamental because it reposes on hashes of all underlying file-blocks. Therefore, it allows efficient and secure verification of data content.

4.3 Multi-Agent Systems (MAS)

The MAS is a set of intelligent agents that interact with each other or with their environments in order to resolve a problem or achieve an objective by using the resources and the knowledge of each agent. An agent is considered as a computer system situated in an environment where it acts in an autonomous and flexible way to achieve the goals for which it was designed (Figure 5).

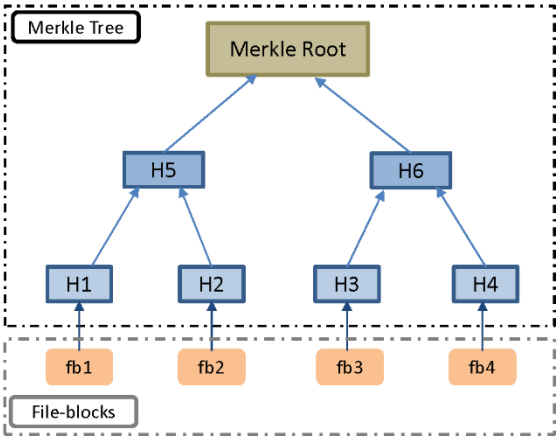

Figure 4. Merkle Hash Tree

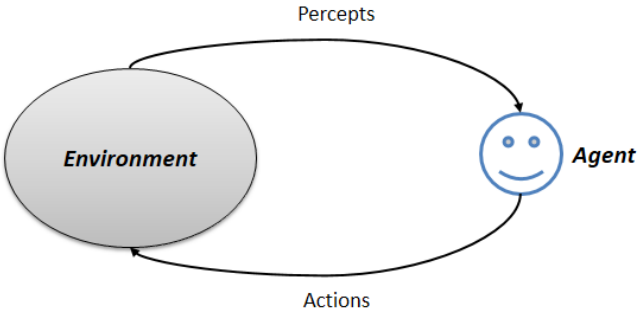

Figure 5. An agent in its environment

Intelligent agent percepts dynamic conditions in its environment; it acts to affect conditions in its environment; and it reasons to interpret perceptions, solve problems, draw inferences, and determine actions[25].

Agents enjoy the following properties[26]:

- **Autonomy**: agents are able to operate without any intervention of humans or others, they have control over the action to undertake among those that are possible;
- **Reactivity**: agents perceive their environment and respond quickly to changes occurred;
- **Pro-activity**: agents are able to manifest objective-directed behavior by taking initiatives.
- **Social**: agents are able to interact with other agents via agreed languages such as Knowledge Query Manipulation Language (KQML) or Agent Communication Language (ACL).

The figure 6 shows the interaction between agents and their environment. Agent organizations are based on an organizational paradigm similar to that of human organizations. In fact, one of the objectives of the MAS is to adapt and assimilate patterns of human organizations into computer systems in order to better follow their actual behaviors. This last point makes the MAS well adapted to the problems of modeling and simulation of organizations. Depending on the scope given to a system, agents can be designed under different specific organizations that are basing on defined rules and structures (Figure 6). These organizations are described as hierarchies, holarchies (holon organizations), coalitions, teams, congregations, societies, federations and matrix organizations[27].

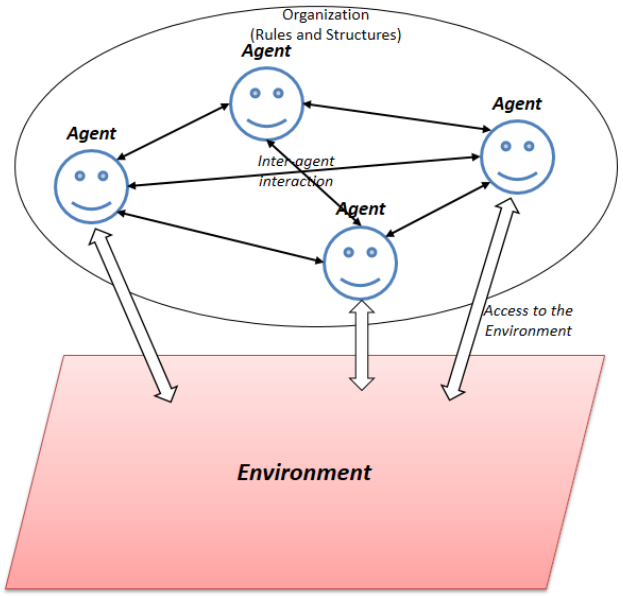

Figure 6. Agents and there organizational relationships

It is complicated for a single agent to game a system. Thereupon, Multi-agent systems were invented to solve problems that are difficult or impossible for an individual agent. Recently, many researchers have proposed multi-agent systems combined with cloud computing environments using knowledge bases for managing the storage of cloud client's data.

4.4 Deduplication

Data deduplication (called also intelligent compression or single-instance storage) permits to keep from storing multiple copies of the same data that allows reducing data volumes thereby reduces storage overhead. Data deduplication has been broadly applied to save storage overhead in the cloud server. There are two methods for performing data deduplication, both of them use the same process to identify redundant data, the difference resides in the location of the deduplication processing:

- **Client-side data Deduplication**: called source-based deduplication, it is the elimination of redundancies from data before transmission to the backup target. It uses the client software for comparing new data block with the previously backed up data blocks before storing this new one in the storage server, which saves storage space and bandwidth.
- **Server-side data Deduplication**: called also target-based deduplication, it is the removal of redundancies from data after their transfer to storage server. In this type only storage space is saved[28].

5. Description of our Proposed Approach

5.1 The Proposed Approach

The authors devoted this section to the description of their proposal. When a cloud user tries to add a file to the cloud server, the CSP checks the existence of the entire file or some of its file-blocks in the storage server. The reason behind this working manner is to reduce the amount of stored data in the cloud server storage. Our approach uses Blockchain data structure, where each block contains information for a file of a user. Each block contains the user ID (U_{id}), the file ID (F_{id}), version number v, timestamp t, the number of file-blocks N, the Merkle Tree and the hash of the previous block in the chain. One block may correspond to the file F1 of user U1 and the next block may correspond to the fileF2 of user U2. The main goal of using Blockchain is to ensure integrity of client's data. The figure 7 presents the architecture of the proposed approach.

For the sake of clarity, the authors begin with the case where a file will be stored for the first time. This case includes a file-blocks level deduplication on the file while comparing these file-blocks with other file-blocks stored previously in order to maintain only the unduplicated ones. Then, the authors extend this case to where the same file will be stored by the same or other cloud user.

Case1: Storing a file for the first time

-- The Client sends the file

--CSP divides the file into N file-blocks (fb_0, fb_1, … fb_{N-1}) where $N=2^d$ and d is the depth of the Merkle tree, then he calculates its hashes $h(fb_i)$,where $0 \leq i \leq N-1$, in order to compute the Merkle Root n_0. Thereafter, he computes the Merkle Root n_0(for i in N−2....0 : $n_i=h(n_{2i+1}\|n_{2i+2})$) of the file. Then the CSP stores the file-blocks in a temporary folder.

--CSP checks if $n_0$already exists in the Merkle Roots Database, if it does not exist, the CSP stores the Root n_0 with the client ID in the Merkle Root Database.

--CSP performs a file-blocks level deduplication on the file by comparing the calculated hashes with the ones located in the hash database,

Case1-1: Storing all file-blocks of a file

--If he did not find any identical file-blocks, the CSP stores the hashes of all the file-blocks in the hash database

--The CSP stores the file-blocks in the storage server and he destroys them from the temporary folder

--The CSP initiates data information: U_{id}, F_{id}, v, t, N corresponding to the file, then he creates a new block in the Blockchain, this block contains file information and the Merkle Tree that correspond to the file.

--The CSP sends a pointer to the client.

In this case, the CSP should store all the file-blocks of a file because he did not find any duplicated file-blocks in the storage server.

Case1-2: Storing some file-blocks of a file

--If he finds some identical hashes, the CSP ignores these hashes then he stores the other file-blocks hashes (unduplicated hashes) in the hash database.

--The CSP stores the other file-blocks (unduplicated file-blocks) in the storage server and he destroys all file-blocks from the temporary folder

--After that, the CSP initiates data information: U_{id}, F_{id}, v, t, N corresponding to the file, then he creates a new block in the Blockchain, this block contains file information and the Merkle Tree that correspond to the file.

--The CSP sends a pointer to the client.

In this case, the CSP does not need to store all the file-blocks because some of them are stored previously by the same or other users. So, CSP ignores the duplicated file-blocks, which reduces disk utilization.

Case2: Storing the same file

--The client sends the file

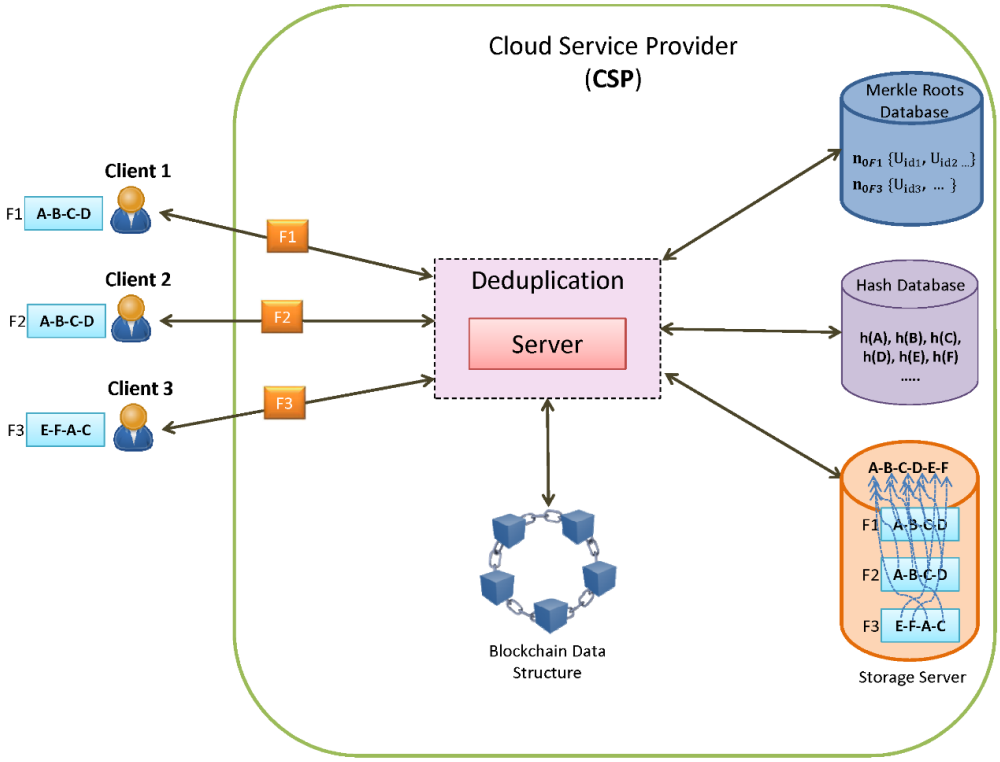

Figure 7. Architecture of the proposed approach

--CSP divides the file into N file-blocks and stores them in a temporary folder, then he calculates the Merkle Root n_0
--CSP checks if n_0 already exists in the Merkle Roots Database, if it exists, the CSP verifies if the actual client ID exists in the ID list that corresponds to this Root

Case2-1: Storing the same file by the same user
--If the actual client ID exists, so the CSP destroys the file-blocks from the temporary folder and he informs the data owner that he has previously stored this same file.
In this case, the CSP does not need to store the file because the same user has stored it previously which reduces the disk's utilization.

Case2-2: Storing the same file by another user
--If the actual client ID does not exist in the ID list, so the CSP adds the client ID to this list, then he destroys the file-blocks from the temporary folder.
--After that, the CSP initiates data information: U_{id}, F_{id}, v, t, N corresponding to the file, then he creates a new block in the Blockchain, this block contains file information and the Merkle Tree that correspond to the file.
--The CSP sends a pointer to the client.
In this case, the CSP does not need to store the file because another user has stored it previously which reduces the disk's utilization.

5.2 MAS to ensure Deduplication Technique

In this work, the authors apply data deduplication in the cloud server side (Figure 8), where they use MAS for modeling several autonomous intelligent agents: Interface Agent, Mediator Agent, Analysis Agent, Control Agent and Data agent. Each agent hassome specific tasks to achieve, starting with receiving files, going through data deduplication and ending with storing these files. The combination of the whole tasks of each agent will produce the main objective of the proposed approach.

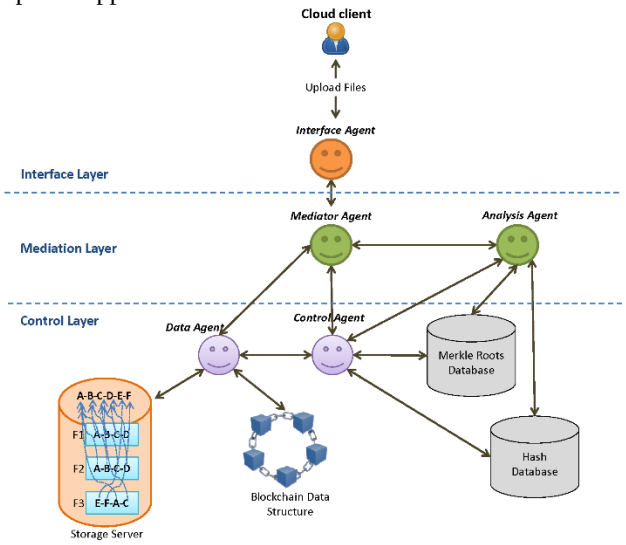

Figure 8. Architecture of the proposed multi-agent system model

Interface agent
The rule of interface agent is to interact with users for receiving files and transmitting them to the Mediator agent for further use by other agents.

Mediator agent
The main function of this agent is to manage the communication between agents. It is responsible for transferring the files and sending back the pointers to the Interface agent.

Analysis agent
The objective of this agent is to check if the Merkle Root and the client ID exist into Merkle Roots Database in order to

identify duplicated files. He is also responsible for performing a file-blocks level deduplication on the file by comparing the calculated hashes with the ones located in the hash database in favor of singling out duplicated file-blocks.

Control agent

This is the agent who computes the Merkle Root of a file and sends it tothe Analysis agent to be compared. He also stores the file-blocks hashes in the hash database and the Root with the client ID in the Merkle Root Database in order to use them in the next storage operations.

Data agent

Data agent has two main tasks to accomplish; the first one is to create a new block in the Blockchain, this block contains the Merkle Tree and information corresponding to the file. The second one is to store the file-blocks corresponding to that file in the storage server.

5.3 Sequence Diagrams of MAS based Deduplication

The execution of the proposed model for possible cases is illustrated in the following sequence diagrams. The distribution of tasks and interactions between agents shows the interest of the use of MAS to ensure good governance of the storage service.

Case1: Storing a file for the first time

--The cloud client uploads a file
--The Interface agent transmits the file to the Mediator agent
--The Mediator agent sends the file to the Control agent
--The Control agent divides the file into N file-blocks then he computes the Merkle Root $n_0$of the file. After that, the Control agent stores the file-blocks in a temporary folder.
--The Control agent sends the client ID with the Root n_0 and the hashes of file-blocks to the Analysis agent
--The Analysis agent checks if n_0 already exists in the Merkle Roots Database, if it does not exist, he performs a file-blocks level deduplication on the file by comparing the calculated hashes with the hashes located in the hash database, then he informs Control agent by the result obtained,

Case1-1: Storing all file-blocks of a file

--If there are no identical file-blocks, the Control agent stores the hashes of all the file-blocks in the hash database then he stores $n_0$with the client ID in the Merkle Root Database.
--Afterwards, the Control agent demands to the Data agent to store all the file-blocks
--The Data agent stores all the file-blocks in the storage server
--Then, the Data agent creates a new block in the Blockchain, this block contains the Merkle Tree and information corresponding to the file.
--The Control agent destroys the file-blocks from the temporary folder
--The Data agent sends to Mediator agent a pointer to the block corresponding to that file
--The Mediator Agent sends to the Interface Agent the pointer
--The Interface Agent retransmits the pointer to the concerned cloud user
--The user preserves a file ID list; this list contains pointers, each pointer directly points to the particular block in the Blockchain corresponding to that file.

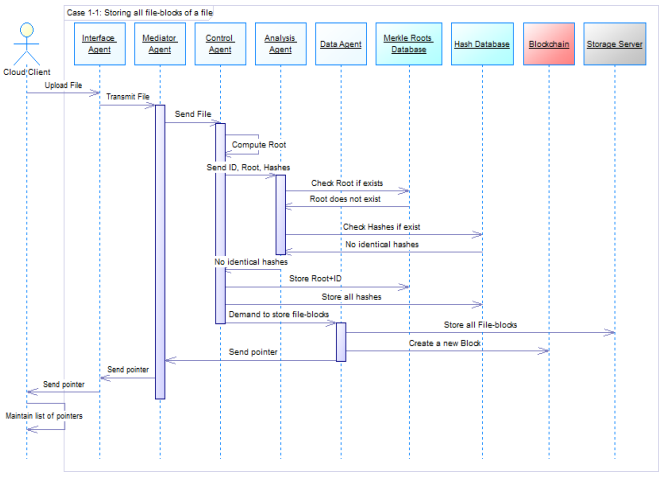

Figure 9. Sequence diagram of case1-1

Case1-2: Storing some file-blocks of a file

--If there are some identical hashes, the Control agent ignores these hashes then he stores the hashes of the other file-blocks (unduplicated hashes) in the hash database, then he stores $n_0$with the client ID in the Merkle Root Database.
--Afterwards, the Control agent demands to Data agent to store the unduplicated file-blocks
--The Data agent stores the other file-blocks (unduplicated file-blocks) in the storage server
--Then, the Data agent creates a new block in the Blockchain, this block contains the Merkle Tree and information corresponding to the file.
--Control agent destroys the file-blocks from the temporary folder
--Data agent sends to the Mediator agent a pointer to the block corresponding to that file

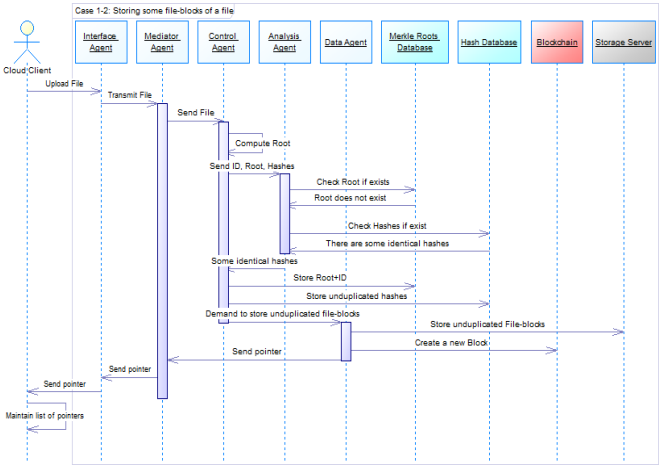

Figure 10. Sequence diagram of case1-2

--The Mediator Agent sends to the Interface Agent the pointer
--The Interface Agent retransmits the pointer to the concerned cloud user
--The user preserves a file ID list; this list contains pointers, each pointer directly points to the particular block in the Blockchain corresponding to that file.

Case2: Storing the same file

--The cloud client uploads a file
--The Interface agent transmits the file to the Mediator agent
--The Mediator agent sends the file to the Control agent
--The Control agent divides the file into N file-blocks then he computes the Merkle Root $n_0$of the file. After that, the Control

agent stores the file-blocks in a temporary folder.
--The Control agent sends the client ID with the Root n_0 and the hashes of file-blocks to theAnalysis agent
--The Analysis agent checks if n_0 already exists in the Merkle Roots Database, if it exists, the Analysis agent verifies if the actual client ID exists in the ID list that corresponds to this Root, then he informs the Control agent by the result obtained,

Case2-1: Storing the same file by the same user
--If the actual client ID exists, the Control agent destroys the file-blocks from the temporary folder and he informs the Mediator agent that the file exists for the same cloud user,
--The Mediator agent transmits the result to the Interface agent
--The Interface agent informs the cloud client that he had previously stored this same file.

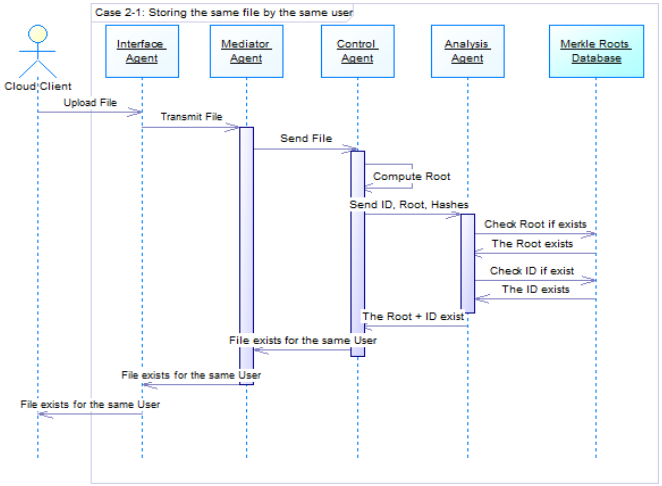

Figure 11. Sequence diagram of case2-1

Case2-2: Storing the same file by another user
--If the actual client ID does not exist in the ID list, the Control agent adds the client ID to this list
--After that, the Control agent destroys the file-blocks from the temporary folder then he asksthe Data agent to create a block in the Blockchain
--The Data agent creates a new block in the Blockchain, this block contains the Merkle Tree and information corresponding to the file.
--The Data agent sends to Mediator agent a pointer to the block corresponding to that file
--The Mediator Agent sends to the Interface Agent the pointer
--The Interface Agent retransmits the pointer to the concerned cloud user
--The user preserves a file ID list; this list contains pointers, each pointer directly points to the particular block in the Blockchain corresponding to that file.

5.4 Data Integrity Auditing

The auditing tasks are delegated to an external competent entity that is the Third Party Auditor. The auditing requests are sent directly by users to TPA. Besides, TPA could perform multiple auditing tasks, in a simultaneous way, received from different users. TPA is semi-trusted and allowed to verify the integrity of files, but it is prohibited to have access to the content of these files. CSP is semi-trusted and allowed to see the content of data, but it is obliged to follow the steps needed for the auditing process. The figure 13 shows the system model for public auditing where the three entities interact with each other.

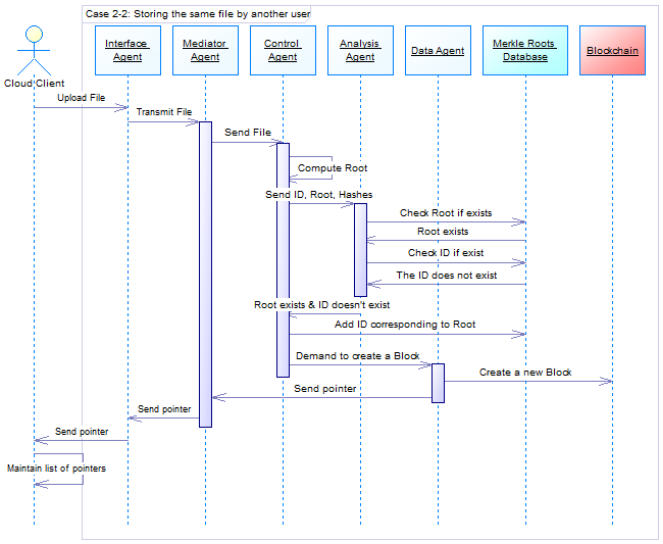

Figure 12. Sequence diagram of case2-2

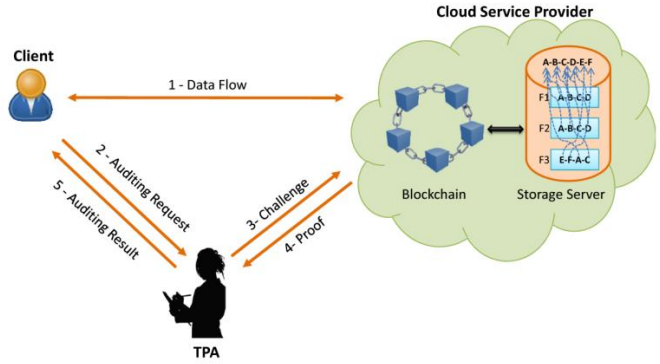

Figure 13. System model for public auditing

In this approach, the authors use Blockchain to store information of data in order to preserve data integrity in Cloud Computing. In the following, the authors present the auditing process. They refer to the technique of verification used in [29] and they make some refinement:
--TPA asks CSP for v and t that correspond to the file to verify
--CSP sends v and t corresponding to the file
--TPA computes the generator seed $r = h^P(n_0)$ where leaves are divided into P chunks
--After that, TPA derives the leaf numbers in each P chunk as: for j in 0 ... P – 1: $l_j = G(r, j)$ with G some cryptographic pseudo-random number generator (PRNG)
--Then, TPA sends the leaf numbers $\{l_j\}$ to the CSP
--CSP provides the appropriate sibling information to the TPA, which allows the TPA to compute the new Merkle root n'_0
--TPA verifies if $n_0 = n'_0$ or not
--TPA then calculates the new seed $r' = h^P(n'_0)$
--TPA deduces the leaf numbers $l'_j = G(r', j)$
--Hereupon, the TPA checks whether $l'_j = l_j$ for each j in $0 \ldots P-1$, hence, if they match, then the file is verified.
--Finally, the TPA informs the user by the result obtained.
The sequence diagram is shown in figure 14.

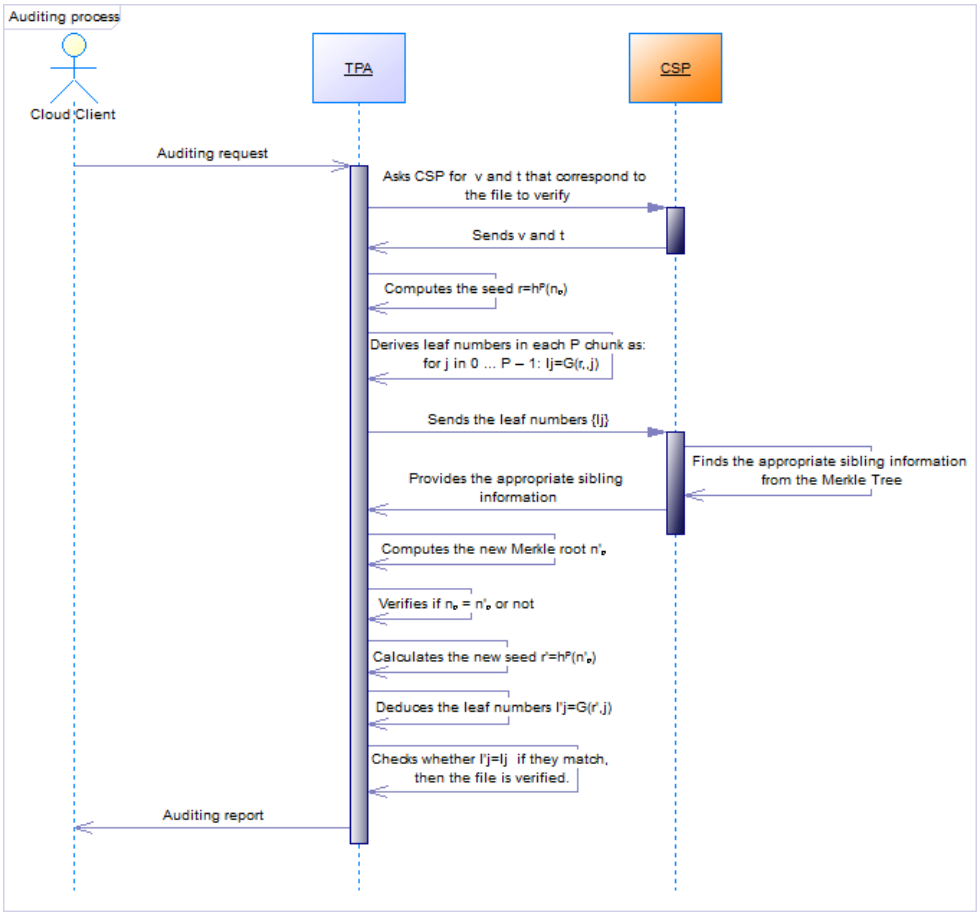

Figure 14. Sequence diagram for auditing a file

6. Discussion and Analysis

To achieve the goals cited above, our method relies on the combination of two techniques that are Blockchain and deduplication, in order to achieve both data integrity and storage efficiency. As shown in our approach, when there is no replica in the file-blocks, the CSP needs to store all the file-blocks of the file in the storage server. However if there are some duplicated file-blocks, the CSP needs to store just the unduplicated file-blocks which reduces disk utilization. In other way, if a user tries to store the same data, CSP informs him that he had already stored it in a previously storage operation. According to our model, the same file-block is stored only once on the cloud storage server in order to save storage space. The authors can consider data deduplication as an ideal method to eliminate redundant data and minimize storage and network overhead. The Multi-agents system is characterized by adaptability, cooperation and distribution that permit to deal with the evolution of Cloud computing in an efficient and controlled manner. Therefore, as shown in our approach, multi-agents systems are well adapted to manage efficiently deduplication in Cloud computing.

On the other side, according to the technique that the authors used in the auditing process, TPA will have no idea about owners' data, which implies that confidentiality of data is ensured against auditors. Besides, TPA could perform multiple auditing tasks, in a simultaneous way, received from different users. In the case where the TPA receives several auditing requests for the same file derived from different users, it may be ineffective to handle them as individual tasks rather than batching them together and performing only one audit task by interacting the CSP to check the data integrity. After that, it replies all concerned users by the auditing result. Hence the deduplication technique is not only efficient for data storage, but it is also efficient for the auditing process where multiple users want to verify the same file which allows to reduce the communication and computation cost between the auditor and CSP. Furthermore, if a small bit is changed in any file-block, it will be certainly a significant difference in the resulting Merkle root, which confirms that the Blockchain data structure is a good choice to preserve integrity of clients' data.

7. Conclusion and Ongoing Works

In this paper, the authors presented a new promising approach, for efficient managing of data deduplication in the Cloud server-side and ensuring data integrity auditing, based on the Multi-agent System and Blockchain technologies. Deduplication involves a lot of computation to eliminate the redundant data. To solve such problem, the authors integrate in their proposal a multi-agent system where five intelligent agents are working in cooperation to manage the complexity of the deduplication process. This process has been broadly applied to save storage overhead in the cloud server.

The authors have also extended their method by using the Blockchain data structure which ensures data integrity. The public auditing process is manipulated by a third party auditor who checks the correctness of the outsourced data. The main goal for using the Blockchain is that TPA can verify the integrity of data without learning any knowledge of user's

data. The interest of using deduplication has also been shown in the auditing process where several users wish to audit the same file which allows to reduce communication and computation cost between the auditor and CSP.

The authors have demonstrated the feasibility of using the Multi-agent System and Blockchain technologies to manage deduplication and auditing processes through the description of their proposed model. Now, the authors are looking for the implementation of the proposed approach, with lightweight technologies adapted to cloud environment by using the Java Agent Development framework (JADE) which allows the development of Multi-agent Systems, in order to compare it to the existing schemes.

References

[1] NIST, "NIST Definition of Cloud Computing," *The National Institute of Standards and Technology (NIST)*, 2016. [Online]. Available: http://www.nist.gov/itl/cloud/.

[2] O. Achbarou, "A New Distributed Intrusion Detection System Based on Multi-Agent System for Cloud Environment," *IJCNIS*, vol. 10, no. 3, pp. 526–533, 2018.

[3] G. Ateniese *et al.*, "Provable data possession at untrusted stores," *Proc. 14th ACM Conf. Comput. Commun. Secur. - CCS '07*, p. 598, 2007.

[4] G. Ateniese *et al.*, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1–34, 2011.

[5] A. . Juels and B. S. . Kaliski Jr., "Pors: Proofs of retrievability for large files," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 584–597, 2007.

[6] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, no. 3, pp. 442–483, 2013.

[7] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, pp. 1–29, 2015.

[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, 2011.

[9] C. Liu *et al.*, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, 2014.

[10] Y. Zhang and M. Blanton, "Efficient Dynamic Provable Possession of Remote Data via Update Trees," *ACM Trans. Storage*, vol. 12, no. 2, pp. 1–45, 2016.

[11] Y. Zhu, G. J. Ahn, H. Hu, S. S. Yau, H. G. An, and C. J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 227–238, 2013.

[12] H. Tian *et al.*, "Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 701–714, 2017.

[13] C. ming Tang and X. jun Zhang, "A new publicly verifiable data possession on remote storage," *J. Supercomput.*, pp. 1–15, 2015.

[14] A. Li, S. Tan, and Y. Jia, "A method for achieving provable data integrity in cloud computing," *J. Supercomput.*, pp. 1–17, 2016.

[15] Y. Yu *et al.*, "Cloud data integrity checking with an identity-based auditing mechanism from RSA," *Futur. Gener. Comput. Syst.*, vol. 62, pp. 85–91, 2016.

[16] Z. Xu, L. Wu, M. K. Khan, K. K. R. Choo, and D. He, "A secure and efficient public auditing scheme using RSA algorithm for cloud storage," *J. Supercomput.*, vol. 73, no. 12, pp. 5285–5309, 2017.

[17] K. M. Lee, K. M. Lee, and S. H. Lee, "Remote data integrity check for remotely acquired and stored stream data," *J. Supercomput.*, vol. 74, no. 3, pp. 1182–1201, 2018.

[18] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," 2012, p. 1. Association for Computing Machinery (ACM). https://doi.org/10.1145/2133601.2133603

[19] J. Li, X. Chen, F. Xhafa, and L. Barolli, "Journal of Computer and System Sciences Secure deduplication storage systems supporting keyword search," *J. Comput. Syst. Sci.*, vol. 1, pp. 1–10, 2015.

[20] M. Miao, J. Wang, H. Li, and X. Chen, "Secure multi-server-aided data deduplication in cloud computing," *Pervasive Mob. Comput.*, vol. 24, pp. 129–137, 2015.

[21] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *2013 IEEE Conference on Communications and Network Security, CNS 2013*, 2013, pp. 145–153.

[22] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Www.Bitcoin.Org*, p. 9, 2008.

[23] L. Er-rajy and O. El Kiram My Ahmed; El Ghazouani, Mohamed; Achbarou, "Blockchain : Bitcoin Wallet Cryptography Security , Challenges And Countermeasures," *J. Internet Bank. Commer.*, vol. 22, no. 3, 2017.

[24] Merkle, R.C.: Secrecy, Authentication, and Public Key Systems. PhD thesis, Stanford University, Technical Report 1979-1; Information Systems Laboratory (June 1979)

[25] B. Hayes-Roth, "An architecture for adaptive intelligent systems," *Artif. Intell.*, vol. 72, no. 1–2, pp. 329–365, 1995.

[26] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," The Knowledge Engineering Review, vol. 10, no. 2, pp. 115–152, 1995.

[27] B. Horling, and V. Lesser, "A survey of multi-agent organizational paradigms," *The Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, 2004.

[28] L. González-Manzano and A. Orfila, "An efficient confidentiality-preserving Proof of Ownership for deduplication," *J. Netw. Comput. Appl.*, vol. 50, pp. 49–59, 2015.

[29] F. Coelho, "An (almost) constant-effort solution-verification proof-of-work protocol based on Merkle trees," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5023 LNCS, pp. 80–93, 2008.