Proceedings of the Third International Conference on Smart Systems and Inventive Technology (ICSSIT 2020)
IEEE Xplore Part Number: CFP20P17-ART; ISBN: 978-1-7281-5821-1

Data Auditing in Cloud Storage using Smart Contract

Lekshmi M M, Narayanan Subramanian
Center for Cyber Security Systems and Networks
Amrita Vishwa Vidyapeetham
Amritapuri, India
lekshmimm@am.students.amrita.edu, nsubramanian@am.amrita.edu

***Abstract*—In general, Cloud storage is considered as a distributed model. Here, the data is usually stored on remote servers to properly maintain, back up and make it accessible to clients over a network, whenever required. Cloud storage providers keep the data and processes to oversee it on capacity servers based on secure virtualization methods. A security framework is proposed for auditing the cloud data, which makes use of the proposed blockchain technology. This ensures to efficiently maintain the data integrity. The blockchain structure inspects the mutation of operational information and thereby ensures the data security. Usually, the data auditing scheme is widely used in a Third Party Auditor (TPA), which is a centralized entity that the client is forced to trust, even if the credibility is not guaranteed. To avoid the participation of TPA, a decentralised scheme is suggested, where it uses a smart contract for auditing the cloud data. The working of smart contracts is based on blockchain. Ethereum is used to deploy a smart contract thereby eliminating the need of a foreign source in the data auditing process.**

***Index Terms*—Cloud storage, TPA, blockchain, smart contract.**

I. Introduction

The excess amount of data generated across the globe needs an efficient management, which includes its storage and confidential safeguarding. In this technological era, the enormous amount of locally storing data seems to be impossible. This is where cloud storage finds its applications. According to National Institute of Standards and Technology (NIST) Information Technology Laboratory, cloud computing is defined as follows: [1] "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable and reliable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal consumer management effort or service provider interaction."

In order to store and access data, cloud storage provider (CSP) offers a remote storage. The two different cloud computing models are - Service model and Deployment model. Public, private, community and hybrid belong to the former model whereas, software, infrastructure and platform are latte. Google Drive, Carbonite, Dropbox are some of the CSPs available. The users can add or retrieve their data as and when required. The data centers used by cloud service providers for storage are distributed systems susceptible to technical failures. Hardware techniques like RAID and ECC memory and software techniques like erasure memory and replication are used to avoid corruption. But still problems arise. The client requires the data on the cloud to stay safe.

Minimal expense of communication over a high bandwidth network to cloud enables the access to a plethora of IT resources that endures an immense level of utilization, which remains as the most important economic justification for the usage of cloud computing. The cloud remains capable of expanding or reducing resources allocated to it quickly and efficiently and thereby satisfies the requirements of self-service cloud computing feature. The cloud storage architecture is shown in Fig. 1.

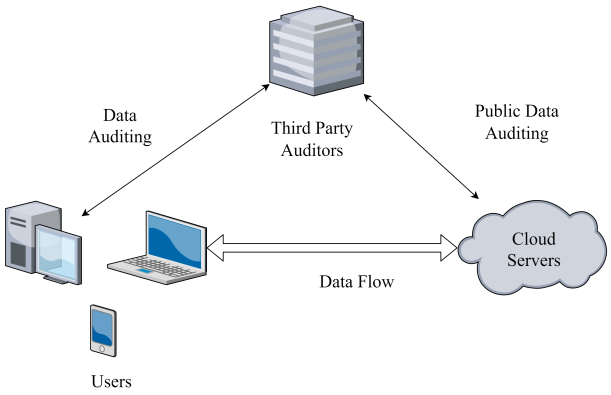

Fig. 1. Cloud storage architecture

Apart from being an abode of massive data storage, it also poses several security concerns. Some of the main issues being data privacy, absence of automatic backup, and data leakage. In addition to this, there also exist dangers that put forward by Advanced Persistent Threats (APTs), incomplete data deletion and many more. The cloud being an insecure space, it requires an auditing scheme for protecting information. As cloud storage services are prone to corruption, a major proportion of companies are expected to opt out of using them in the coming years. This necessitates a new audit scheme with a trusted entity.

TPA which is a centralized unit that performs off-site data integrity auditing is the most commonly used one. Due to the indulgence of a third party other than the client and CSP, the integrity of data remains a question mark. There is no guaranteed security from the side of TPA regarding

data confidentiality as it can be manipulated. A decentralised entity like block chain is a solution that addresses these issues. Satoshi Nakamoto first put forward the notion of blockchain as a fundamental technology that enables the operation of digital assets like cryptocurrency and bitcoin [2] [3].

II. RELATED WORK

A. Third Party Auditor

The data is sent by the cloud users to the cloud service providers over a network. The data can be confidential information, the user's personal information, passwords of personal accounts or bank information. Secure socket layer, VPN are used to secure transactions. Intruder attacks on security services have occurred many often. Avoiding malicious attacks when data transfer occurs between the cloud service provider and users is not always possible. Users require legal assurance on their data security. In this case a third party based authentication system is necessary. It monitors the transaction between the user and CSP. Clients and CSP function on a legal Service Level Agreement (SLA). In case if any data error occurs then the CSP hides it from the client. To avoid this type of issue TPA is used [4]. TPA must be aware of SLA between the CSP and the user and serve as a trusted entity amidst them. The integrity of cloud data can be verified by TPA, but at times the privacy of the cloud users is compromised.

Third Party Auditor (TPA) regularly verifies user data integrity. The three entities involved are a client, TPA and cloud server. The client splits the file to blocks, encrypts them, generates a hash value for each encrypted block. Then it performs hash value concatenation and signature generation on it. TPA performs similar activities like hash value generation, concatenation and signature generation on blocks retrieved from a cloud server. TPA compares it's signatures to verify whether it matches the one created by the client. A difference in the signatures indicates that the data stored on the cloud is tampered. Encryption, integrity check and digital signature calculation are performed by AES algorithm, SHA-2 and RSA respectively. Fig. 5 explains the working of a TPA.

B. Smart Contarct and Blockchain

A blockchain is a series of unchangeable data, maintained by computers that are not under a single centralised source [5]. Cryptographic principles keep the blocks of data secure and maintain the blocks bound to each other. A hash pointer links each block. In a blockchain, data gets generated in successive blocks. A piece of data contained in the data block verifies content from the preceding block. If in case the previous block is modified, the successive blocks don't match up. Attempts to modify data are identified and blocked by the system that manages the blockchain, and hence it is free from tamper [6].

Smart contracts are protocols associated with blockchain that serve as a computer code agreement between two parties [7]. Transaction between two agents that distrust each other enables the elimination of a third party, thereby provides security and reduces transaction cost. The transactions are completely dependent on the agreement. In order to execute

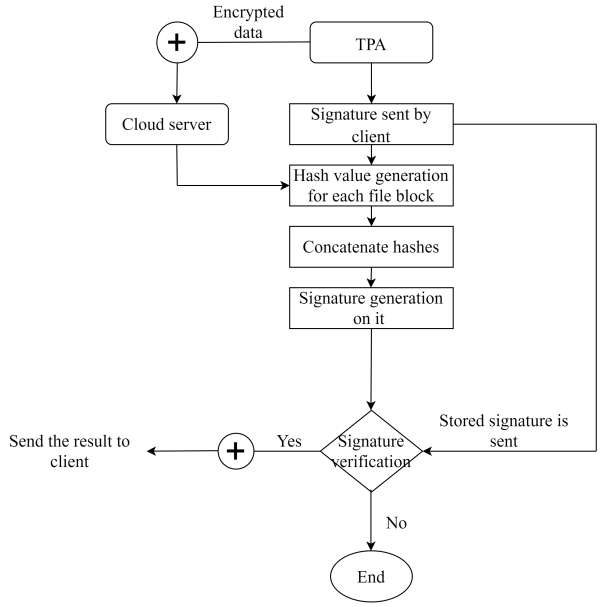

Fig. 2. Flowchart for working of TPA

the agreement of terms stated in a smart contract, the code within which permissions and information are present requires a particular set of events to occur. Smart contract contains the blockchain's security coding. Thus preventing the execution of smart contracts is impossible [8].

Ethereum is an ecosystem based on blockchain that features smart contracts. EVM bytecode or Solidity is used to write the contracts. The coding of smart contracts requires an intense level internal data organisation. Execution of the smart contracts in Ethereum takes place at Ethereum Virtual Machine or EVM, which holds all Ethereum network nodes connected together [9]. EVM performs the compilation and execution of the smart contract code. Truffle is a popular development framework of Ethereum used to create and then deploy a basic solidity template.

A high level programming language called Solidity is used for developing the contract. It has a JavaScript-like syntax. It supports inheritance and polymorphism in addition to user-defined types and libraries [10]. Contract code comprises variables and functions that modify the contract structure. The special variables defined by Solidity hold properties necessary to access invocation-transaction information and blockchain. Solidity has an added advantage of enclosed code units called modifiers that enable to alter the flow of executing code. It works in accordance with condition- oriented programming which eliminates the function body's conditional paths. Modifiers can effortlessly change the function behaviour by mentioning them after the function name in a list. Conditions are checked prior to execution by using modifiers.

Ganache tool facilitates the running of a simulated full-node Ethereum client on our machine and interaction with the contract. We have to install Ganache and Truffle. Smart contracts can be developed after installation. To run the smart

contract we develop a simulated environment. The Ganache tool by default creates 10 accounts each of which has 100 ethers used for testing. The 10 accounts are linked with 10 private keys [11].

III. Proposed System

To eliminate the TPA in cloud storage, we propose a decentralised auditing scheme. Based on the smart contract, we can check the data integrity, thereby preventing the unwanted interferece of data owners and data users. The proposed system is aimed at verifying the accuracy of cloud data at preset time intervals. Thereby the data leakage to third parties at the time of auditing can be effectively curbed.

The system comprises a data owner, a CSP and a blockchain. The data stored in the cloud can be text, images, audio and video. Before storing data into the cloud, the owner encrypts data. The files are splitted into blocks and for each block a hash value is generated. Ethereum smart contract is used for smart auditing in a platform that is decentralized. The smart contract dias holds the contract submitted by the owner, which includes information regarding the file such as name, size, hash and data owner details. CSP then verifies the data integrity. Fig,3 shows the system model.

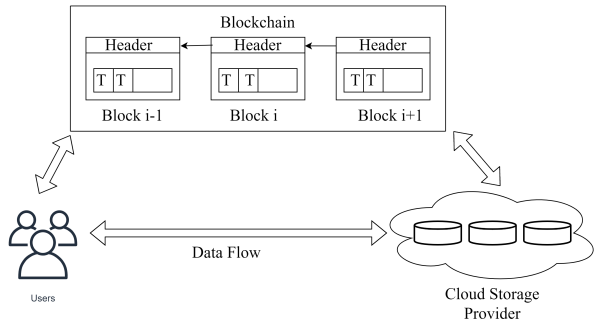

Fig. 3. Proposed system model

An optimized instantiation, the Data Auditing Blockchain (DAB) has been designed for cloud data auditing. It depends on a growing block list that are chained within themselves. A block constitutes a header and relevant auditing proofs. Block header consists of timestamp, previous blocks' hash and Merkel root. Merkle hash tree contains all auditing proofs and transactions. Similarly,blocks store all auditing proofs and Merkle root guarantees their integrity. Fig 4 is an illustration of DAB.

Our scheme uses the Java programming language. The system used is Intel core i3 processor which runs at 2.33 GHz and 4 GB RAM.

A. Preliminaries

Let G and G_T denote two multiplicative groups having generator g and order q. Let $e : G \times G \rightarrow G_T$ be a bilinear map defined on it. The properties of e are :

- Bilinear: For $u, v \in G$ and $a, b \in Z_q^\star, e(u^a, v^b) = e(u, v)^a b$

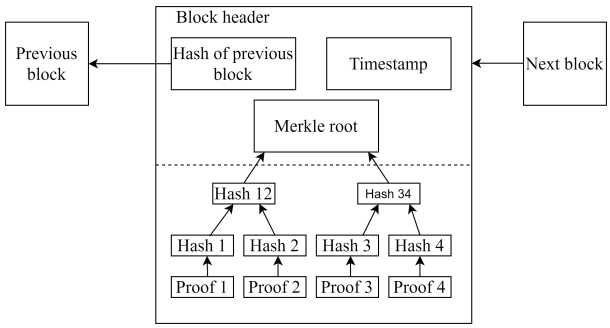

Fig. 4. Data Auditing Blockchain

- Non-degenerate: $\exists g_1, g_2 \in G$ such that $e(g_1, g_2) \neq 1$
- Computable: An efficient algorithm computes map e

Consider $H : \{0,1\}^\star \rightarrow G$, where H is a hash function that maps a string. Secret key $x \epsilon Z_q^\star$ and public key $Y = g^x$ are generated by each data owner.

To begin with, data blocks $b_1, b_2, ...b_n$ are generated by the data owner. A tag is generated for each data block b_i as

$$t1 = \left(h \left(ID_i || i \cdot \prod_j =_1 u_j^{b_{ij}} \right) \right)^x \tag{1}$$

For this, a local encryption library is used. ID_i is an identifier that is unique to the block of data used. Then, the blocks of data and their respective tags are loaded to CSP. On data reception, a smart contract is signed with data owner by the CSP, in order to make an agreement regarding parameters like auditing frequency and range, data corruption penalty and so on. When the smart contract is invoked, random generators are utilized. It has a unique seed for generation of indexes and coefficients of data blocks which are challenged. These parameters are sent to the CSP by the smart contract for integrity check of the data blocks in CSP. Then CSP accesses the challenged data blocks present in its storage servers. At last, the aggregated data block μ_j and tag t are sent by CSP to the smart contract through blockchain.

$$\mu_j = \sum_{(i,v_i) \epsilon Q} V_i \cdot b_{ij} \tag{2}$$

$$t = \prod_{(i,v_i) \epsilon Q} t_i^{v_i} \tag{3}$$

Here V_i is the coefficient. Q is the challenge set which includes index and corresponding coefficient of each and every challenged data block. To conclude, aggregated data block j and tag t is sent by CSP to the smart contract through the blockchain. On reception of the aggregated data block and tag, hash value

$$h_1 = h\left(ID, ||i\right) \tag{4}$$

for each data block is computed by the smart contract. Then, it verifies the integrity of checked data blocks using

$$e\left(t, g\right) = e\left(\prod_{(i,v_i) \epsilon Q} h_i^{v_i} \prod_{j=1} u_j^{\mu_j}, Y \right) \tag{5}$$

In case if (5) holds, 1 is reported by the smart contract to the data owner, or else 0.+

IV. Performance Analysis

The decentralised nature of the proposed protocol makes auditing confidential and reliable. Public auditing has been effectively performed by enabling both the owner and the user of the data to sign the smart contract with the concerned cloud service provider. This ensures that the integrity of data is preserved. Both the auditing of protocol and verification of data block are performed automatically as the execution of the protocol by the smart contract occurs on behalf of the client and server. As a result, once a contract is signed, the indulgence of client or server is not required. The ethereum performance can be understood from Fig 5.

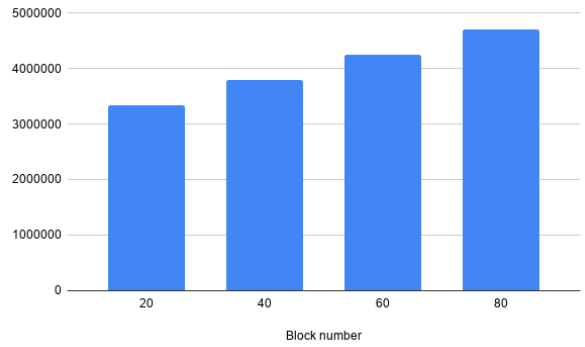

Fig. 5. Gas of verification of smart contract

V. Conclusion

In this paper, a blockchain based decentralized and smart system for auditing cloud storage has been proposed. In auditing, a centralized system like TPA was replaced by a smart contract that enables automatic auditing. The data stored in the cloud by the client is checked at regular intervals in the presence of a client. This eliminates the need for periodic examination by the client. As the storage and execution of smart contracts take place in all network nodes, the result of auditing remains tamper free. The protocol is also efficiently accomplished.

References

[1] R. L. Krutz and R. D. Vines, Cloud security: a comprehensive guide to secure cloud computing. New Delhi: Wiley, 2016.
[2] C. Li, J. Hu, K. Zhou, Y. Wang, and H. Deng, “Using Blockchain for Data Auditing in Cloud Storage,” Cloud Computing and Security Lecture Notes in Computer Science, pp. 335–345, 2018.
[3] N Satoshi, ”Bitcoin A Peer-to-Peer Electronic Cash System,” White Paper 2008
[4] S. More and S. Chaudhari, “Third Party Public Auditing Scheme for Cloud Storage,” Procedia Computer Science, vol. 79, pp. 69–76, 2016.
[5] N. Chowdhury, “Introduction to Blockchain,” Inside Blockchain, Bitcoin, and Cryptocurrencies, pp. 3–26, 2019.
[6] I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev, and L. Yalansky, “Ensuring data integrity using blockchain technology,” 2017 20th Conference of Open Innovations Association (FRUCT), 2017.
[7] “Smart Contracts,” 2019.
[8] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making Smart Contracts Smarter,” Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.
[9] W.-M. Lee, “Understanding Blockchain,” Beginning Ethereum Smart Contracts Programming, pp. 1–23, 2019.
[10] R. Modi, Solidity programming essentials: a beginners guide to build smart contracts for Ethereum and blockchain. Beginners guide to build smart contracts for Ethereum and blockchain: Packt, 2018.
[11] C. Dannen, Introducing ethereum and solidity: foundations of cryptocurrency and blockchain programming for beginners. New York: Apress, 2017.